Table of Contents

Ventrilo 4 Theme Development.

Version 4.0.0 (Tweaks 12/13/2017 Rev-D)

Be sure to read the Ventrilo Theme Guidelines after reading this document for ideas on how to make an appealing theme.

All themes are distributed as Ventrilo Theme Pack (VTP) files. But theme designers build and edit themes using regular system folders.

When editing, every theme must be in a unique folder at the following folder location.

$(Documents)\VentriloData\Themes

For example, the “default” theme would be a folder called

$(Documents)\VentriloData\Themes\default

A theme can also be a VTP File (Ventrilo Theme Package). For example:

$(Documents)\VentriloData\Themes\Default.vtp

The $(Documents) is shorthand for your documents folder and does not translate into any kind of system environment variable. It is for documentation purposes only. This is unique for each login and for each platform. An easy way to open this folder is to right click in the Ventrilo main window, select View then select Working Directory. The following is an example of what the folder path names might look like on different platforms.

Win-XP	= %USERPROFILE%\My Documents
Win-Vista and newer	= %USERPROFILE%\Documents
Mac	= $HOME/Documents

We recommend that a theme designer consider using an alternate configuration to distinguish between development and production versions. This way you can keep your normal installation of the Ventrilo client and use it just like anyone else would based solely on VTP files and using the directory structure mentioned above. But your development environment would be using a dedicated folder structure that is different from the above and would be based on folders rather than VTP’s. The upside to this is that you can use your production version to test your VTP before you release it to other people. See the section further down called Production vs. Development environments.

Each theme must have a file in the folder or VTP named “settings.ini”. This file is used to describe how the theme appears and how certain windows are displayed.

Do not modify the default theme. Any changes to it will be overridden by the program at start up. It is recommend that you download one of the custom Themes from the Ventrilo.com website and use it to learn how a theme works.

Production vs. Development environments:

We encourage all Theme Designers to run two separate configuration environments for your Ventrilo client. One for Production and one for Development.

Your production environment would have the Theme Designer mode turned off which implies that it will only read VTP files and function exactly like a normal Ventrilo Client. This is a good way to test final production builds of your theme when packed into a VTP.

Your development environment would have the Theme Designer mode turned on which enables reading configuration and image files from ordinary folders. In this mode your log file will show potential errors in the scripts and other problems such as missing files. The production version would not show these kind of things.

We strongly recommend that you don't use the same name between production and development environments when logging in to the same Ventrilo server. This would confuse other users who might need to talk to you or send you files. For example:

Production login name = JohnSmith
Development login name = JohnSmith-Dev

The Production environment is considered to be the default installation of the Ventrilo client and using the default configuration folder. This is typically the $(DOCUMENTS)\VentriloData folder. See notes at the start of this file describing this.

A Development environment is used by appending a –W (and additional info following it if desired) to a command line option that instructs Ventrilo to use a completely different path for ALL of its configuration data. While the VentriloData folder will always exist the –W option allows you to specify where that folder is placed. This is essentially overriding the "Working Directory" that you would see if you had right-clicked -> View -> Working Directory in the Ventrilo main window.

Creating an alternate launch icon to start Development environment.

Microsoft Windows:

1. Select the VentriloPro Icon on your Desktop.

2. Press Ctrl-C and then press Ctrl-V.

3. A new VentriloPro shortcut will now be displayed on your desktop with a modified name.

4. Right click on the new icon and select "Properties".

4. Append a –W (dash w) to the end of the Target field outside of the quotes surrounding the path and executable name. There must be a space after the trailing quote but no space between the dash and the w.

Example: `"C:\Program Files\VentriloPro\Ventrilo.exe" –w`

5. Remove everything from the "Start In" field if not already empty.

6. Click on the General Tab and rename the shortcut.

Example: `VentriloPro Development`

7. Click OK to save your changes.

When you double click on the new icon it will create a VentriloData folder on your desktop where you can create your development version of themes.

If you would like the VentriloData folder to be someplace else then enter an existing folder name in the "Start In" part of the window mentioned above. Such as: %USERPROFILE% would create the folder here, where YourLoginName is your Windows login account name.

`C:\Users\YourLoginName\VentriloData`

Apple Macintosh:

Download the "VentriloWork.dmg" launch scripts for Mac. These were created to provide the same basic functionality as mentioned above for the Windows platform. Due to the fact that the Macintosh does not allow for creating shortcuts with launch parameters like Windows does, we created these launchers using AppleScript to accomplish the same thing.

The DMG file contains the following scripts including the source code for each one. Select one of the files that shows a Scroll icon that matches your desired method of running your dev environment and copy it to the desktop. You can now double click the icon on your desktop to start your development environment.

The ".scpt" version of each file is the associated scripts source code. If you desire you can customize the script. Copy the script source code file to your desktop and then double click on it. This will start the AppleScript Editor. The source code for each script contains comments on how to turn them into actual executable script applications and possible warnings.

If you click on the Dock bar "Launch" icon and then click on VentriloPro that should be considered your production environment. Clicking on one of these scripts via your desktop would be your Development environment.

The following describes how each particular script will function.

Ventrilo Work Desktop

This will place the VentriloData folder on your desktop.

Ventrilo Work Home

This will place the VentriloData folder in your $HOME directory.

To locate the VentriloData folder for this script in the Finder follow these steps:

Click Desktop
Press Cmd-N
Press Shift-Cmd-H for your home directory.

Ventrilo Work Preferences

This will place the VentriloData folder in your $HOME/Library/Preferences directory.

By default the Mac Finder will not display your personal version of the Library folder. You can make it visible with the following steps:

Click Desktop
Press Cmd-N
Press Shift-Cmd-H for your home directory.
Right click in Finder window and select “Show View Options”
In the tool window that opens check the “Show Library Folder”
Double click on the now visible “Library” folder.
Double click on the “Preferences” folder.

Ventrilo Work Embedded

This will place the VentriloData folder inside the script that you double clicked on. Hence embedding the data within the script itself. Beware, if you recompile the script the embedded VentriloData folder will be automatically destroyed. This is caused by the AppleScript Editor and not the script itself.

To locate the embedded VentriloData folder right click on the script you’re using and select “Show Package Contents”.

Settings.ini file

How to use

An INI file is composed of two types of entries, Topics and Keys. A Topic is a name inside of brackets like the following:

[MainBorder]

Inside each topic there will be one or more Keys with an associated value. A Key / Value pair will look like this:

ColorActive = 0,0,0

The Key part is “ColorActive” whereas the Value part is the “0,0,0”

Each key is unique to the specified topic name that it follows. The same key name can exist in multiple topics for this reason. The above example is an actual key/value from an actual topic named [MainBorder]. But it can also be used in the topic [ChannelBorder]. So don’t be surprised if you see a key name used multiple times inside the same settings file.

However, the same key name can NOT be used in the same Topic. If you do have duplicate key names in the same topic the last instance of the key is the one that will be used. Comment out or delete unused key names to prevent confusion about which one will be used.

If you want to create a test version of the key you can place a // in front of the original key to prevent it from being used.

Comments can be placed on the key/value line by using a double slash // as part of the string. The // and everything following it is ignored. Example:

ColorActive = 0,0,0 // Border color for active window.

Key / Value pairs follow certain rules. For example a key that starts with Img (short for Image) specifies an image file name such as JPG or PNG file. But it can also contain Properties that instruct the program on how the image should be used. For example:

[MainBorder]

ImgLeft = mbi-left.png, Bottom, Repeat 3

The first part of the value specifies the file name “mbi-left.png”. The property “Bottom” means that the image should be displayed and aligned to the bottom of the window. The property “Repeat 3” means that the image should be repeated upwards 3 times.

The “Repeat 3” is an example of a property that also has its own optional parameter.

It is important to note that properties have different meanings depending on the key they are attached to. In the above example the key being used is ImgLeft in a Border topic. This key is defining the image to be displayed for the windows left side border. As mentioned in this case the Bottom property instructs the program to display the image starting from the bottom. The Repeat 3 tells it to replicate

the image 3 times moving towards the top of the window but on the left side. Had the property been "Repeat" without the number 3 it would have meant to repeat the image so as to cover the entirety of the left side of the window.

This sounds harder than it is but with this document you should have a good idea of how and when keys and properties are used.

In this document you will occasionally see the following as the default value for some of the key names. These names are not macro values that you can use. They are for documentation purposes only and in case the values change in a future version of the program.

(Default not used)	Indicates the key is ignored unless specified.
(System setting)	Indicates the value comes from the Operating System.

Properties

The following is a list of all property names. Keep in mind their meaning can change depending on what topic and key they are being used with. See the section "Properties explained" for details about each property.

Left
Top
Right
Bottom
Center
Align
Under
Clip
NoClip
Tile
MatchBorder
Stretch
Aspect
NoPressHover
StateImages
Erase
Alpha #
Repeat (#)
Ofs # #

VertPara # # #
HorzPara # # #

Colors

Keys that start with the tag “Color” use the following two formats.

Color = R,G,B

Color= A,R,G,B

The RGB letters stand for Red, Green and Blue. Each value is between 0 and 255 where 0 is no color and 255 is the brightest value for that color component.

When entered as ARGB the A stands for Alpha also known as transparency. When the alpha is less than 255 the color that is painted for the given key will be blended with the background of what is being painted over. 0 is totally transparent and 255 means totally opaque. If you are using an Alpha of 255 it is best just to use a 3 part R,G,B definition and let the program assume the color is opaque, doing so could also translate to increased performance when drawing.

Never use 0,0,0 due to the irregular border requirement that transparent parts of the border image use the true black value of 0,0,0,0. You should avoid having any part of your theme use this value. We recommend that the darkest value to use should be 5,5,5. Preferably you should use the program provided macro $(DEFBLACK). This will prevent a perceived black section from becoming transparent (i.e. see thru). It also provides some color mapping space for anti-aliasing of fonts when drawing on a supposed black background. The average user will not notice the difference between the two values and most monitors won’t display a noticeable difference.

Macro string substitution

The theme settings.ini file supports macro string substitution, the ability to define a string once and have it used multiple times. For example, you want to predefine the background color for all parts of the main window. You must have the [Macros] topic and a key name of your choosing.

```
[Macros]
BLACK                = $(DEFBLACK)
WHITE                = $(DEFWHITE)
COLORBACKGROUND      = $(BLACK)
```

You could then use that key in place of literal values throughout the remainder of the INI file like this:

```
[MainTop]
ColorBackground = $(COLORBACKGROUND)

[MainToolbar]
ColorBackground =  $(COLORBACKGROUND)

[MainTree]
ColorBackground =  $(COLORBACKGROUND)

[MainMute]
ColorBackground =  $(COLORBACKGROUND)
```

The above example would make the background color for all parts of the main window the same color. You could then tweak the macro's value to be 0,0,128 and the entire window would turn a shade of blue.

You can also use macros to replace parts of a value:

```
[Macros]
BORDERFOLDER          = small

[MainBorder]
ImgLeft          = $(BORDERFOLDER)/mbi-left.png
```

The above example would allow you to switch between multiple folders of images to do comparisons, or say one is a production copy and the other is an "in development" copy.

We also recommend that you create macros to predefine color values that must be handled with care. This would be useful should color constraints change in future versions. For example:

```
[Macros]
BLACK   = $(DEFBLACK)
WHITE   = $(DEFWHITE)
HOVER   = $(DEFHOVER)
ICONBACKGROUND = $(DEFICONBACKGROUND)
COLORNOTUSED = $(DEFCOLORNOTUSED)
```

The program creates the following default macros. It is strongly recommended that you not use them directly but instead create your own variant macro as demonstrated in the above example. This way you can change the appearance of the window simply by changing a single macro.

```
DEFBLACK              = 5,5,5

DEFWHITE              = 250,250,250

DEFGRAY               = 128,128,128
```

DEFHOVER = 35, 155, 226

DEFICONBACKGROUND = 144, 137, 120

DEFCOLORNOTUSED = 0,0,0,1

Sequentially numbered key names

Some key names can be entered multiple times in the same topic, but they have sequential numbers appended to them or they use the Auto Numbering character to do the numbering for you. The following example key is from the [MainBorder] and [ChannelBorder] topics. It allows multiple offset overlays to be used thru out the specified window.

ImgOverlayOfs = mbo-cornertopright.png, top, right, Ofs -35x-22, erase
ImgOverlayOfs1 = mbo-cornertopleft.png, top, left, Ofs -22x-22, erase
ImgOverlayOfs2 = mbo-cornerbottomright.png, bottom, right, Ofs -35x-35, erase

The problem with manually encoding the numbers is that once a break in the sequence is found the program will stop reading the remainder of the numbered keys of the same name.

An easier way of doing this is to forgo the literal numbering of each key, and instead append the # sign to the end of the key name. The program will see this and automatically append the next sequence number to the key name.

ImgOverlayOfs# (First one found will translate to ImgOverlayOfs)
ImgOverlayOfs# (Second one found will translate to ImgOverlayOfs1)
ImgOverlayOfs# (Third one found will translate to ImgOverlayOfs2)

There are several advantages to using the Auto Numbering character. 1) It's less tedious, 2) You can copy and paste without worrying about conflicting numbers, 3) You can change the order that overlays are drawn simply but moving them up or down inside the topic, 4) You can add, delete or comment one out without being forced to renumber the list.

This technique can be used on certain keys such as the example above, and others like ImgOverlayInternal. See documentation of each key where applicable.

Predefined topic names

At the time of this writing the Ventrilo client will look for the following hard coded Topic names:

[Macros]
[General]
[Main]
[MainBorder]
[MainTop]
[MainToolbar]
[MainTree]
[MainMute]
[Channel]
[ChannelBorder]
[ChannelTree]
[SideCtrlPanel]
[PrivateChat]
[PrivateChatBorder]
[Equalizer]
[EqualizerBorder]
[SbcMain]
[BtnMain]
[CmbMain]
[ChkMain]
[ChkMuteSnd]
[ChkMuteMic]
[ChkMsg]
[ChkPTT]
[EditScroll]
[EditInput]
[Slider]

It is possible to create your own custom Topic names in order to make the settings file easier to read. Examples on how and why you would do this will be discussed further down in this document.

The following is a detailed description of each hard coded topic name.

[General]

This topic describes keys that are unique to the theme and of general use.

MinVers = 4.0.0

Tells the program that for this theme to run correctly the client program must be version X.Y.Z or newer. When the client running the theme is less than the specified version it will pop up a warning message telling the user that this version or newer is required. The user can chose to continue using the theme or switch to the built-in Default theme. If left blank the theme assumes version 4.0.0

[Main]

This topic describes keys that are unique to the Ventrilo main window.

Multiple [Main] topics can be created to give the end user multiple choices of how the window appears. This is done by appending a number to the topic name such as [Main1], [Main2], [Main3]. This will display 3 additional appearance modes in addition the default [Main] option. The valid range of additional appearances is between [Main1] and [Main255].

Numbered versions of the [MainTop], [MainToolbar], [MainTree], [MainBorder] and [MainMute] can exist as well.

SizeDividerHorz = 1
SizeDividerVert = 1

These options control the size of the divider that surrounds the tree part of the window in relation to the content part of the window. The larger the number for the given direction will push the tree inwards. These options only apply when the border is enabled. See the NoBorder variant below for when it is not enabled. Advanced themes usually set these values to 0.

SizeDividerHorzNoBorder = 1
SizeDividerVertNoBorder = 1

These options control the size of the divider that surrounds the tree part of the window in relation to the content part of the window. The larger the number for the given direction will push the tree inwards. These options only apply when the border has been disabled by the user. See the variants above for when the user has enabled the border. Advanced themes usually set these values to 0.

ColorDivider = $(DEFBLACK)

This option specifies the color to be drawn for the SizeDivider options above.

Name = (Default not used)

This option tells the program the name of the appearance when the user right clicks in the window and selects the “Appearance” menu option. If not specified the program will use the topic name which is not very informative to the end user. It is highly recommended that you provide a human readable Name. Can be used in [Main] and [Channel] variants. If you do specify a name it must be unique when compared with the other appearance names you might create for Main / Channel windows.

UseBorder = MainBorder

Specifies the topic name to use for this windows border. Ex: UseBorder=MainBorderAlt. Implies you have a topic named [MainBorderAlt].

[MainBorder]

BlendDesktop = 1

Optional: Instructs the window to draw everything so that it can alpha blend with the desktop if the colors and graphics are used accordingly. Setting this value to 0 will disable blending with the desktop and will disable support for irregular shaped windows. The default theme turns this feature off.

An irregular border graphic that has an alpha channel as part of its graphic will blend seamlessly with the desktop background information. Interior parts of the window can also be set to draw using alpha channel information making them semi-transparent with the desktop. Example: [MainTree] ColorBackground = 192,5,5,5

SizeLeft = (System setting)

Forces the left side border to be X number of pixels wide overriding the system setting. If a graphic is used as the border then the larger of the two will decide the actual border width.

Example: SizeLeft = 10

SizeRight = (System setting)

Forces the right side border to be X number of pixels wide overriding the system setting. If a graphic is used as the border then the larger of the two will decide the actual border width.

Example: SizeRight = 10

SizeTop = (System setting)

Forces the top side border to be Y number of pixels tall overriding the system setting. If a graphic is used as the border then the larger of the two will decide the height.

Example: SizeTop = 10

SizeBottom = (System setting)

Forces the bottom side border to be Y number of pixels tall overriding the system setting. If a graphic is used as the border then the larger of the two will decide the height.

Example: SizeBottom = 10

SCPShiftLeft = 0

Value range: -256 to 256

When a Side Control Panel is displayed on the Left side of the window this value will shift horizontally the SCP by X number of pixels. If the value is negative it shifts the SCP to the left, otherwise it shifts to the right.

SCPShiftRight = 0

Value range: -256 to 256

When a Side Control Panel is displayed on the Right side of the window this value will shift horizontally the SCP by X number of pixels. If the value is negative it shifts the SCP to the left, otherwise it shifts to the right.

MinHorz = 0

Specifies the minimum horizontal pixel size for the content part of the window. This is useful to prevent the user from shrinking the window horizontally causing the top and bottom graphics of the border to be clipped. By doing so you are imposing a minimum window width that the user might not like. Comment out this key to turn it off rather than setting it to 0.

MinVert = 0

Specifies the minimum vertical pixel size for the content part of the window. This is useful to prevent the user from shrinking the window vertically causing the left and right graphics of the border to be clipped. By doing so you are imposing a minimum window height that the user might not like. Comment out this key to turn it off rather than setting it to 0.

IncHorz = 0

Specifies the horizontal increment in pixels that a window will be forced to when sizing the left or right border. This is useful if you have a top or bottom border image that replicates and you want the graphic to properly line up with the corner graphic making it appear to be seamless.

IncVert = 0

Specifies the vertical increment in pixels that a window will be forced to when sizing the top or bottom border. This is useful if you have a left or right border image that replicates and you want the graphic to properly line up with the corner graphic making it appear to be seamless.

MaxHorz = 0

Specifies the maximum horizontal pixel size for the content part of the window when enabled. Useful when used in conjunction with the [Channel] ArtMode key. When 0, blank or not specified the feature is disabled. Setting the value to -1 enables the option but is translated into a max value of 0.

MaxVert = 0

Specifies the maximum vertical pixel size for the content part of the window when enabled. Useful when used in conjunction with the [Channel] ArtMode key. When 0, blank or not specified the feature is disabled. Setting the value to -1 enables the option but is translated into a max value of 0.

TbbFolder = TbbMain

Specifies the folder to use for the TitleBar Button images. If an alternate is specified the folder must exist, otherwise the titlebar buttons will not be displayed. If you wish to override the default buttons in your theme simply create a folder under your theme called “TbbMain” and place the appropriate files in there and leave this option unchanged. The only time you would specify a different folder name is if one of your theme borders will use different titlebar buttons simultaneously. For Example, you want the TBB’s in your channel windows to be different from the main window. Note: This could be confusing to the end user if they are looking at different TBB images between windows. Generally a theme would use the same type of buttons across all window borders.

TbbForceSide =

This option will force the TitleBar buttons to always be on a specific side of the window. This can be useful when border graphics that are near the top of the window and on a specific side and you don’t want the buttons to obscure the graphics. Keep in mind the Mac displays the buttons on the left while Windows and other platforms generally display on the right by default, and some users might not like having the buttons on the wrong side of the window for their specific platform.

When set to Left or Right the program will ignore the platform default and the theme designer Flip option. If not specified the program will display the buttons on the side of the window as dictated by the platform or according the theme designer Flip option.

Properties:

Left = Force buttons to left side of the window.
Right = Force buttons to right side of the window.

TbbProp = StateImages

Specifies properties that should be applied to the TitleBar Buttons.

Properties:

NoPressHover = Disables the hover state when the button is pressed.
StateImages = Forces all states to use unique single image. No overlays.

TitleIndentButtons = 0

Applies to titlebar buttons when displayed on RIGHT side of window. Specifies a horizontal indent from the right side of the title bar min/max/close buttons. This is useful if you have an ImgOverlayInternal graphic that would be displayed in the same area of the window as these buttons.

TitleIndentButtonsLeft = 0

Applies to titlebar buttons when displayed on LEFT side of window. Specifies a horizontal indent from the left side of the title bar min/max/close buttons. This is useful if you have an ImgOverlayInternal graphic that would be displayed in the same area of the window as these buttons.

TitleSpaceButtons = 5

Specifies the spacing between the title bar buttons. If you create custom versions of these buttons you might decide the spacing should be different or even be 0.

TitleFillAfter = 0

When set to 1 instructs the titlebar to fill the background after any overlay has been painted instead of before. This is useful for assuring that the active and/or inactive background color is visible when using a full background overlay that covers the entire content area of the window. Best used when ColorTitleActive and ColorTitleInactive options are configured to use an alpha color component.

ColorTitleActive = (System setting)
ColorTitleInactive = (System setting)

These options control background color of the Title bar when the window has focus or when it no longer has focus.

ColorTitleText = (System setting)

This option controls the color of the text in the Title bar.

ColorActive = (System setting)
ColorInactive = (System setting)

These two options will control the color of the border when the window is active or inactive, if the window is using a simple rectangle as the border.

ImgTitleMin = tbb-min
ImgTitleMax = tbb-max
ImgTitleWindow = tbb-window
ImgTitleClose = tbb-close

These options allow the designer to override the base names of the graphic files that will be used to draw the title bar Minimize, Maximize, Windowed (Restore) and Close buttons. Each base name will have one of several variant names appended to them. For example:

tbb-min-act.png
tbb-min-inactive.png
tbb-min-disable.png
tbb-min-hover.png
tbb-min-press.png

It is best to leave these options alone. Instead you can use the TbbFolder option to create alternate title bar buttons but using the same naming convention for the individual files.

Folder = (Default not used)

Instructs all Img files in this border topic to read from a sub-folder of the current theme from the specified folder name. If not specified the files will be read from the theme folder itself. Note: The ImgTitle keys mentioned above are exempt from this Folder definition as of the time of this writing.

Every window has two types of borders depending on the window state of Active (Focus) or Inactive (No Focus). The following keys and their associated values and properties each have the same meaning but will be used according to the window state. The key that does not have the word “Inact” in its name is the Active state version of the key.

ImgLeft =
ImgLeftInact =

These keys specify the image file to use for drawing the left side of the border.

Properties:

Top = Align image to top side of content.
Bottom = Align image to bottom side of content.
Center = Center image between top / bottom sides of content.
Repeat = Repeat image to other side of content. See optional parameters.

ImgRight =
ImgRightInact =

These keys specify the image file to use for drawing the right side of the border.

Properties:

Top = Align image to top side of content.
Bottom = Align image to bottom side of content.
Center = Center image between top / bottom sides of content.
Repeat = Repeat image to other side of content. See optional parameters.

ImgTop =
ImgTopInact =

These keys specify the image file to use for drawing the top side of the border.

Properties:

Left = Align image to left side of content.
Right = Align image to right side of content.
Center = Center image between left / right sides of content.
Repeat = Repeat image to other side of content. See optional parameters.

ImgBottom =
ImgBottomInact =

These keys specify the image file to use for drawing the bottom side of the border.

Properties:

Left = Align image to left side of content.
Right = Align image to right side of content.
Center = Center image between left / right sides of content.
Repeat = Repeat image to other side of content. See optional parameters.

ImgOverlayLeft =
ImgOverlayLeftInact =

These keys specify the image file to use for overlaying the left side of the border.

Properties:

Top = Align image to top side of content.
Bottom = Align image to bottom side of content.
Center = Center image between top / bottom sides of content.
Under = Draw the overlay under the main border image.
Clip = Prevent the overlay from extending past the content area.
Erase = Clears the background to transparent where the overlay will be drawn.

ImgOverlayRight =
ImgOverlayRightInact =

These keys specify the image file to use for overlaying the right side of the border.

Properties:

Top = Align image to top side of content.
Bottom = Align image to bottom side of content.
Center = Center image between top / bottom sides of content.
Under = Draw the overlay under the main border image.
Clip = Prevent the overlay from extending past the content area.
Erase = Clears the background to transparent where the overlay will be drawn.

ImgOverlayTop =
ImgOverlayTopInact =

These keys specify the image file to use for overlaying the top side of the border.

Properties:

Left = Align image to left side of content.
Right = Align image to right side of content.
Center = Center image between left / right sides of content.
Under = Draw the overlay under the main border image.
Clip = Prevent the overlay from extending past the content area.
Erase = Clears the background to transparent where the overlay will be drawn.

ImgOverlayBottom =
ImgOverlayBottomInact =

These keys specify the image file to use for overlaying the bottom side of the border.

Properties:

Left = Align image to left side of content.
Right = Align image to right side of content.
Center = Center image between left / right sides of content.
Under = Draw the overlay under the main border image.
Clip = Prevent the overlay from extending past the content area.
Erase = Clears the background to transparent where the overlay will be drawn.

ImgOverlayBackground =
ImgOverlayBackgroundInact =

These keys specify a single image that will be displayed in the background covering the entire content area of the window (i.e. non border space).

Properties:

Left = Align image to left side of content.
Right = Align image to right side of content.
Top = Align image to top side of content.
Bottom = Align image to bottom side of content.
Center = Center image between left / right or top / bottom.
Tile = Draw image repeated to covert entire content area.
Aspect = Draw image covering content area while maintaining aspect ratio. Can be combined with Center.
Alpha = Apply an alpha channel to the entire image.

ImgOverlayInternal =
ImgOverlayInternalInact =

These keys specify image files that are position aligned with the overlays of the border but appear inside the content area of the window making it look like the border overlay is going thru the window itself.

Multiple internal overlays can be specified in this topic. See the section named "Sequentially numbered key names".

Properties:

Left = Align image to the left side of content.
Right = Align image to the right side of content.
Top = Align image to the top side of content.
Bottom = Align image to the bottom side of content.

Center = Align image to center of content area. Must include one edge property.
Alpha = Apply an alpha channel to the entire image.
Erase = Clears the background to transparent where the overlay will be drawn.

VertPara = Vertical Parallax. See Properties explained.
HorzPara = Horizontal Parallax. See Properties explained.

ImgOverlayOfs =
ImgOverlayOfsInact =

These keys specify image files that are position aligned to the edges of the windows content area making it look like the border overlay is going thru the window itself. Similar to ImgOverlayInternal except there is no need to break the overlay graphic into multiple parts.

Multiple offset overlays can be specified in this topic. See the section named "Sequentially numbered key names".

If an image is specified as "Left, Top" the overlay will start at the top left corner of the content area and draw across and down. To push the image into the border you would apply a negative number to the Ofs property. Example: "`Left, Top, Ofs -10x-20`". This would move the image 10 pixels to the left into the left edge of the border and 20 pixels up into the top edge of the border. When specifying Right or Bottom the start of the image is on the outside of the content edge and the Ofs property would be used to move the image into the content area, if so desired.

Any part of the image that appears in the border will affect the overall size of the border for the respective edge. If the edges border is bigger than this overlays overhang then there would be no effect on the borders current size.

If you do not provide a specified edge then you must provide the word Center. When doing this the top/left pixel of the image is calculated to the center of the content area. If you want the graphic to appear centered you will need to apply a negative Ofs of half the width and half the height of the image. Ex: If the image is 100x100 then you would need to specify "`Ofs -50x-50`".

Properties:

Left = Align image to the left side of content.
Right = Align image to the right side of content.
Top = Align image to the top side of content.
Bottom = Align image to the bottom side of content.
Center = Align image to center of content area.
Alpha = Apply an alpha channel to the entire image.
Erase = Clears the background to transparent where the overlay will be drawn.
Ofs = #x# offset from the specified edge. Is usually specified using negative numbers.

Multiple overlay images on any given border edge:

Themes also support multiple overlays on all 4 sides. To specify additional overlays you need only append a number to the key name but the numbers must be in sequence. If the sequence is broken the program will stop looking for additional images.

These keys support the "Sequentially numbered key names" mechanism.

For example, to specify 3 overlays on an Active state top border you would do the following:

ImgOverlayTop = mbo-top.png, Left
ImgOverlayTop1 = mbo-top.png, Right
ImgOverlayTop2 = mbo-top.png, Center

The Inactive state border is similar with the number trailing the key name. Using the above example:

ImgOverlayTopInact = mbo-top.png, Left
ImgOverlayTopInact1 = mbo-top.png, Right
ImgOverlayTopInact2 = mbo-top.png, Center

ImgCornerTopLeft =
ImgCornerTopLeftInact =

These keys specify the image file to use for drawing the top left corner of the border.

Properties:

Align = Forces corner to indent to top right side of border image.
= Forces corner to indent to bottom left side of border image.

ImgCornerTopRight =
ImgCornerTopRightInact =

These keys specify the image file to use for drawing the top right corner of the border.

Properties:

Align = Forces corner to indent to top left side of border image.
= Forces corner to indent to bottom right side of border image.

ImgCornerBottomLeft =
ImgCornerBottomLeftInact =

These keys specify the image file to use for drawing the bottom left corner of the border.

Properties:

Align = Forces corner to indent to bottom right side of border image.
= Forces corner to indent to top left side of border image.

ImgCornerBottomRight =
ImgCornerBottomRightInact =

These keys specify the image file to use for drawing the bottom right corner of the border.

Properties:

Align = Forces corner to indent to top right side of border image.
= Forces corner to indent to bottom left side of border image.

ImgSysTray =
ImgSysTrayFlash =

On the Microsoft Windows platform these two options allow the theme designer to override the default icon when the window is minimized to the System Tray. The first one is the normal state and the Flash version is when the icon is flashing between the two in an effort to get the users attention. You can override one or both images. Each file must be a 16x16 color PNG file. For example:

ImgSysTray = SysTray.png
ImgSysTrayFlash = SysTrayFlash.png

You should consider making unique SysTray icons so that if a user has two Main windows open, using different user names, and they are both minimized there is a visual distinction. If each main window was using a different theme then the user could tell at a glance which icon denotes a specific user and connection. This feature can also be used on ChannelBorder windows making it easy for a user with a single Main window but several channel windows and all of them minimize to the SysTray and each one has selected a different Appearance.

ImgDockIcon =
ImgDockIconFlash =

On the Apple Macintosh platform these two options allow the theme designer to override the system default minimize Dock bar icon for the window. If you make an Icon you should also provide the flash version, otherwise the program will alternate between the system standard of a miniature version of the window and the Dock icon. Example

ImgDockIcon = DockIcon.png
ImgDockIconFlash = DockIconFlash.png

Images must be in PNG format and should be 128 x 128 in size. It is strongly recommended that images use transparent backgrounds like other dock bar icons and with a centered image. If the image isn't centered it will look strange when displayed next to other dock bar icons.

For these dock icons to be visible the system must be configured to minimize a window to a separate icon rather that all windows be grouped in the Application icon. To do this you need need to uncheck "System Preferences – Dock – Minimize windows into application icon"

[MainTop]

This topic describes keys that are used to display the top part of the main window. This is the area that has the 3 buttons for User name, Server, Bindings and their associated pull down combo boxes.

ColorBackground = (System setting)

This option specifies a specific background color for the top part of the main window.

BtnTopUsername = BtnMain
BtnTopServer = BtnMain
BtnTopBindings = BtnMain

These fields allow you to redirect their respective buttons to a different topic name. If you create three different buttons that are visually different from each other you need the option to override the folder and other options associated with each button. By default the program will use settings from the [BtnMain] topic. It will do this for all three variations. For example if you override all 3 of them you might use new topic names such as:

BtnTopUsername = Btn1User
BtnTopServer = Btn1Server
BtnTopBindings = Btn1Bind

The example implies that you create the topic names [Btn1User], [Btn1Server] and [Btn1Bind] and each of these topics have options that are described in the [BtnMain] topic of this document.

You can override a single one of these or all three of them. If you design a button that is the same for all three of these fields but you have different graphics for each unique appearance then you will need to use these fields but you need only create one topic name per appearance. For example:

[MainTop1]

BtnTopUsername = Btn1
BtnTopServer = Btn1
BtnTopBindings = Btn1

[MainTop2]

BtnTopUsername = Btn2
BtnTopServer = Btn2
BtnTopBindings = Btn2

The above example shows two different MainTop's (see documentation for creating multiple appearances) and each of the three buttons point to the same button definition for the respective appearance. In this case you would have topics [Btn1] and [Btn2] which use the same options that you would find in the [BtnMain] section of this document. Inside each of these [Btn?] definitions you would probably have a "Folder =" option that would direct the program to read graphics from the specified

folder name in your theme named Btn1 or Btn2. The folder name is up to you but naming the folder the same as the topic override will make your code easier to understand.

CmbTopUsername = CmbMain
CmbTopServer = CmbMain
CmbTopBindings = CmbMain

These fields allow you to redirect their respective combo box (pulldowns) to a different topic name. If you create three different combo boxes that are visually different from each other you need the option to override the folder and other options associated with each combo. By default the program will use settings from the [CmbMain] topic. It will do this for all three combo variations. For example if you override all 3 of them you might use new topic names such as:

CmbTopUsername = Cmb1User
CmbTopServer = Cmb1Server
CmbTopBindings = Cmb1Bind

The example implies that you create the topic names [Cmb1User], [Cmb1Server] and [Cmb1Bind] and each of these topics have options that are described in the [CmbMain] topic of this document.

You can override a single one of these or all three of them. If you design a combo that is the same for all three of these fields but you have different graphics for each unique appearance then you will need to use these fields but you need only create one topic name per appearance. For example:

[MainTop1]

CmbTopUsername = Cmb1
CmbTopServer = Cmb1
CmbTopBindings = Cmb1

[MainTop2]

CmbTopUsername = Cmb2
CmbTopServer = Cmb2
CmbTopBindings = Cmb2

The above example shows two different MainTop's (see documentation for creating multiple appearances) and each of the three combos point to the same combo definition for the respective appearance. In this case you would have topics [Cmb1] and [Cmb2] which use the same options that you would find in the [CmbMain] section of this document. Inside each of these [Cmb?] definitions you would probably have a "Folder =" option that would direct the program to read graphics from the specified folder name in your theme named Cmb1 or Cmb2. The folder name is up to you but naming the folder the same as the topic override will make your code easier to understand.

[MainToolbar]

This topic describes keys that are used to display the toolbar part of the main window.

ColorBackground = (System setting)

This option specifies the background color to be used when drawing the background of the toolbar portion of the window.

ColorHover = $(DEFHOVER)

This option specifies the color to be used when drawing the hover indicator when the mouse cursor is over one of the buttons. It does not apply when the buttons are configured for state image mode.

ColorXmit = $(DEFBLACK)

This option specifies the foreground color of the "—XMIT--" indicator.

ColorXmitBackground = (Default not used)

Needed only when the toolbar is transparent with desktop. Otherwise the text of the XMIT will be difficult to read.

ColorPing = $(DEFBLACK)

This option specifies the foreground color of the "Ping" indicator.

ColorPingBackground = (Default not used)

Needed only when the toolbar is transparent with desktop. Otherwise the text of the PING will be difficult to read.

PtSizeXmit = 10

Deprecated. Will not be supported in future updates.

PtSizePing = 10

Deprecated. Will not be supported in future updates.

MtbProp = StateImages

Specifies properties that should be applied to the Toolbar Buttons.

Properties:

NoPressHover = When button is pressed the Hover component is not displayed.

StateImages = Forces all states to use unique single image. No overlays.

Main Tool Bar icons can be overridden by creating a folder in your theme named “MainToolBar”. The file names should match those in the Default themes MainToolBar folder.

The default for this option is “StateImages”. You can turn StateImages off by setting this value to 0. If you apply any other properties and do not specify StateImages you will still be turning this mode off. It is strongly recommended that you design your theme using StateImages as it will give the artist the most freedom to control how the buttons look for each state.

NoHorzMin = 0

When set to 1 this option will instruct the toolbar to not calculate a horizontal minimum for the main window. If the toolbar is composed of very large images it can force the minimize size of the window to be larger than the user might care for. This option turns off that calculation. Please note that there are other things in the main window which can also affect a horizontal minimum such as, but not limited to, the mute bar and border graphics.

Folder = MainToolbar

This option tells the program the folder name to use when reading the main toolbar button images. If you have multiple Appearances for the main window you might also wish to create different Toolbar buttons for each Appearance. This option allows for overriding the default folder name.

PingShow = 0

This option will force the background color of the Ping field to always be displayed. This is useful when you set the toolbar and the mute bar to be completely transparent and when the user has disabled the title bar and borders and has squished the window down vertically so that only the toolbar and mutebar are visible. The ping background will give the user a place to grab when moving the window around and for right clicking to bring up the menu.

[MainTree]

This topic describes keys that are used to display the actual content tree window of the Ventrilo client. These values are also used in the channel windows tree display.

DefColors = 0

This option specifies what the default colors will be for certain types of text displayed in the tree. It simplifies changing the colors to work on a dark background and could cover a large range of

background colors and background images. The theme designer can still override specific text types as they see fit.

0 = Used for white or brightly colored backgrounds.
1 = Used for black or dark colored backgrounds.

The following items are affected by the "DefColors" options.

ColorFontChannel =

Bright = 0,128,128
Dark = 0,200,200

ColorFontUser =

Bright = $(DEFBLACK)
Dark = $(DEFWHITE)

ColorFontComment =

Bright = 0,0,255
Dark = 100,196,255

ColorFontIntegration =

Bright = 127,0,0
Dark = 240,0,0

ColorFontGuest =

Bright = 255,64,64
Dark = 255,64,64

ColorFontRank =

Bright = 0,128,0
Dark = 0,203,0

ColorFontCommentLink =

Bright = 0,192,192
Dark = 0,210,210

ColorFontFlags =

Bright = 64,128,0
Dark = 158,210,0

ColorFontChannelComment =

Bright = 128,0,255
Dark = 192,0,255

ColorFontChannelCommentLink =

Bright = 220,0,220
Dark = 220,0,220

ColorChanPassword =

Bright = 196,0,0
Dark = 196,0,0

ColorChanUserAuth =

Bright = 128,128,0
Dark = 128,128,0

ColorSelBar =

Bright = 196,196,196
Dark = 128,196,196,196

The following items are NOT affected by the "DefColors" option.

ColorVoiceBeginBG = 255,255,0
ColorVoiceBeginFG = $(DEFBLACK)

These two options control the background and foreground colors when displaying a user name who is talking but you are not currently receiving a voice stream from. This is the Yellow state.

ColorVoiceRecvBG = 0,255,0
ColorVoiceRecvFG = $(DEFBLACK)

These two options control the background and foreground colors when displaying a user name who is talking and you are receiving a voice stream from. This is the Green state.

ColorVoiceStopBG = 128,0,128
ColorVoiceStopFG = 255,255,255

These two options control the background and foreground colors when displaying a user name who you were receiving a voice stream from but has since stopped. This is the Magenta state.

ColorErrBG = 128,0,0
ColorErrFG = 255,255,255

There are times when the program will need to display error information in the tree so that the user knows that something is wrong. It is also used to indicate that the program is in Theme Designer mode and when a new version is available for download.

ColorBackground = (System setting)

This option specifies the background color to be used for drawing the tree users and channel names.

ColorGradient = (Default not used)

When specified it will force the tree background to be drawn as a gradient between the ColorBackground and this color.

GradientDir = 0

When ColorGradient is specified this will determine the direction of the gradient.

0 = Left to Right
1 = Top Left to Lower Right (Slower drawing)
2 = Top to Bottom
3 = Top Right to Lower Left (Slower drawing)

ColorTransBackUser = (Default not used)
ColorTransBackChannel = (Default not used)

These options specify background color to be applied when drawing a user or channel but only that portion of the window that will contain any substance rather than the entire background. They were originally created as a mechanism to help tree items show up on totally transparent backgrounds. They have limited value and we recommend that they not be used. Setting the background to transparent but with a color is a better solution.

ImgBackground =

Specifies an optional background image to be used inside the tree part of the window.

The Left, Right, Top, Bottom and Center can be combined except when they don't make sense. For example it would make no sense to combine Left with Right or Top with Bottom. Instead you

combine a Top/Bottom with a Left/Right. Center is implied for vertical and horizontal until you tell it otherwise. To force the image to the lower right corner of the tree window you would use:

"ImgBackground = filename, Bottom, Right"

Properties:

Tile = Replicate image horizontally and vertically.
Stretch = Display image so as to cover entire tree window with aspect distortion.
Aspect = Display image so as to cover entire tree window but maintain aspect ratio.
Center = Center the image.
Top = Align with top part of tree window.
Bottom = Align with bottom part of tree window.
Left = Align with left edge of tree window.
Right = Align with right edge of tree window.
Clip = When painting background color clip area of bitmap. Decreases flicker.
Alpha = Blends the image with background color using alpha value. Ex: Alpha 128
Under = If scrollbars have Solid=0 this option will force image under scrollbar.
MatchBorder = Align background "Tile" image with border image.

BtnName = BtnMain

Topic name that will dictate how the push button(s) will be displayed.

SbcName = SbcMain

Topic name that will dictate how the scroll bars will be drawn in the tree part of the window. See [SbcMain] further down in this document. If your channel window has a different visual appearance you can point this key to a different [Sbc???] topic so that the scroll bars appear differently compared to the main windows scroll bars. For example, if you created topics such as [ChannelTree], [ChannelTree1], [ChannelTree2], etc, you could override their respective scroll bars here.

ImgChanAuth = tsi-chanauth.png
ImgChanAuthOpen = tsi-chanauthopen.png
ImgChannel = tsi-channel.png
ImgChannelOpen = tsi-channelopen.png
ImgChanPass = tsi-chanpass.png
ImgChanPassOpen = tsi-chanpassopen.png
ImgNetStatus = tsi-netstatus.png
ImgServer = tsi-server.png
ImgVoiceBegin = tsi-voicebegin.png
ImgVoiceRecv = tsi-voicerecv.png
ImgVoiceStop = tsi-voicestop.png

ImgVoiceOff = tsi-voiceoff.png
ImgVoiceChan = tsi-voicechan.png
ImgVoiceGroup = tsi-voicegroup.png
ImgVoiceU2U = tsi-voiceu2u.png
ImgVoiceP2P = tsi-voicep2p.png

Each one of the above Img definitions with file names starting with "tsi-" can be used to override the default file name for the Tree Status Icons. However, it is best to leave these file names as is and create a sub-folder in your theme called "MainTreeIcons" and use the same names as listed above for the respective image files. You should only override the default tree icons for a specific reason. Keep in mind this can create confusion for users that are accustomed to seeing the default icons, and that many users will turn off the icons entirely since they are not as important with the new display capabilities of the program.

Folder = MainTreeIcons

This option tells the program the folder name to use when reading the main tree icon images. If you have multiple Appearances for the main window you might also wish to create different tree icons for each Appearance. This option allows for overriding the default folder name.

[MainMute]

This topic describes keys that are used to display the mute bar part of the Ventrilo client. Each of the mute bar buttons comes with a folder in the default theme of the same name and each folder contains unique graphics for each of the buttons. Each folder also contains a Control.ini file that overrides the default behavior of a check box so that the text part of the check box is not displayed. You can override the graphics and whether or not the text is displayed in your theme. See the standard theme named "Version 3" for an example of how to do this. If you provide your own folder of a mute bar checkbox name listed here you will not need to override the topic name itself.

ColorBackground = (System setting)

Indent = 5

Specifies the pixel indent from the left side of the mute bar where the buttons will start drawing from. Valid range is 0 to 200.

IndentRight = 0

Specifies the pixel indent from the right side of the mute bar where the buttons will start drawing from. At the time of this writing there is only the one button which is the user enabled Menu button. Valid range is 0 to 200.

SpaceButtons = 5

Specifies the pixel spacing between the mute bar buttons. 0 is a valid option.

ChkMuteSnd = ChkMuteSnd

Topic name that will dictate how the “Mute Snd” check box will be displayed.

ChkMuteMic = ChkMuteMic

Topic name that will dictate how the “Mute Mic” check box will be displayed.

ChkMsg = ChkMsg

Topic name that will dictate how the “Msg” check box will be displayed.

ChkPTT = ChkPTT

Topic name that will dictate how the “PTT” check box will be displayed.

[Channel]

This topic is similar to the [Main] topic and uses the exact same keys for the same purposes with the following additional keys. It creates an alternate way of displaying a channel windows contents and the option of using a different border. Multiple [Channel] topics can be created to give the end user multiple choices of how the window appears. This is done by appending a number to the topic name such as [Channel1], [Channel2], [Channel3]. This will display 3 additional appearance modes in addition the default [Main] option. Similar to [Main] the valid range of appearances is [Channel1] thru [Channel255].

The theme designer can also override how the tree contents are displayed by using the same keys from [MainTree] but in corresponding [ChannelTree], [ChannelTree1], [ChannelTree2] topics. If a corresponding [ChannelTree#] does not exist the program will use the values from [MainTree] by default. Unless your background colors for the channel would clash with the foreground text colors it is best to let them use the values from [MainTree].

UseBorder = ChannelBorder

This option instructs the program which [ChannelBorder] to use. If not specified the program will use the [ChannelBorder] if it exists, otherwise it will default to [MainBorder]. If the theme designer chooses to create an alternate border for an Appearance this should point to a corresponding [ChannelBorder] style name. For example: If this topic is called [Channel2] then for ease of readability you can use the topic name of [ChannelBorder2]. This is a suggestion and not a requirement. Feel free to use channel border topic names as you see fit.

ArtMode = 0

This option puts a channel window into a state that has no practical use and is meant for artistic purposes only, hence the name. It will hide the titlebar for this window. Since the title bar is hidden the user must hold down the Control and Shift keys and then left click the mouse to move it around.

Important: Usage is slightly different as already noted by the need to use Control-Shift to move the window, but it also means the only other ways to close this channel window is via the keyboard or the (Windows) Taskbar or possibly clicking on the specific window in the system taskbar and closing it there..

Via keyboard: The user needs to left click on this channel window and then press the appropriate key sequence for the respective platform for closing a window. On MS-Windows this would be Alt-F4 and on the Mac this would be Command-W. The user can still right click on the channel and switch to a different appearance making the title bar accessible again.

Via Windows Taskbar: The user can hover over the taskbar window, right click and then select the Close menu option, or if the system shows a close button in the image they can click that as well. Note: This implies that "Setup / GUI / Shows channel windows in taskbar" option be enabled.

Last resort: If all else fails the user can switch to the Default theme which would make the channel windows behave like normal desktop windows making the close button available, after which the user could switch back to their desired theme.

Properties:

0 = Artistic mode disabled
1 = Artistic mode enabled

[ChannelBorder]

This topic describes the keys used to control how a channel window border is displayed. See the documentation for the [MainBorder] topic as the keys and values used are the same.

[ChannelTree]

This topic describes keys that are used to display the actual content tree of the channel window. If this topic does not exist, and alternate appearance versions such as [ChannelTree1], [ChannelTree2], etc. the program will default to using the [MainTree] topic.

See the documentation above for the [MainTree] topic as keys and their associated values are the same.

[SideCtrlPanel]

This topic describes the keys and values used to control how the Side Control Panels (SCP) are displayed. We strongly recommend that this section not be changed so as not to create confusion from one theme to the next, with the exception of the **Prop** option which can be useful.

ColorBackground = $(DEFBLACK)

This option specifies the background color to be used when the user has decided to use the SCP in a non-transparent mode.

ColorBorder = $(DEFBLACK)

This option specifies the color to be used when drawing a border around an SCP column, but only when the user has configured the SCP to use a non-transparent mode.

ColorHover = $(DEFHOVER)

This option specifies the color to be used when drawing the hover indicator.

ColorCheck = 0,255,0

This option specifies the color to be used when drawing the Check Mark indicator, but only when the user has configured the program to use tick marks instead of overlays.

ColorDisabled = 192,32,32

This option specifies the color to be used when drawing the Disabled indicator, but only when the user has configured the program to use tick marks instead of overlays.

ColorIndBorder = $(DEFBLACK)

This option specifies the color to be drawn around tick mark indicators for Check and Disabled. Setting this value to the macro $(DEFCOLORNOTUSED) will prevent the indicator border from being drawn.

IconAlpha = 255

This option applies an alpha component when drawing the SCP button images. 255 = fully opaque and 0 = totally transparent.

IndDim = 14,14

When the user has configured the SCP to use tick marks this option specifies the (horz,vert) dimensions of the indicator.

IndOfs = 1,1

When the user has configure the SCP to use tick marks this option specifies the (horz,vert) offset from the respective indicator. The Check indicator will be moved 1 pixel right and 1 pixel down. The Disabled indicator will be moved 1 pixel left and 1 pixel down.

IndType = 1

When the user has configure the SCP to use tick marks this option specifies the type of indicator to draw.

1 = Rectangle
2 = Ellipse

Pad = 0,0

This option specifies the (horz,vert) padding that should be added to each SCP button making the total area of each button slightly larger but leaving the button image the same size, effectively creating a larger hover area but also spacing the button farther apart.

Prop =

This option can be useful when you have a graphical border on the top part of a window that is very tall.

Possible values are:

Align = Aligns SCP with title bar.

By default the SCP will align with the top edge of the border. By setting this option to Align will force the SCP's to align with the top part of the content area of the window. By default this would be the Title bar if the user has it enabled.

[Equalizer]

This topic describes the keys and values used to control the display of the Equalizer window. Multiple Equalizer topics can be created by appending a number to the end of the topic. Example: [Equalizer], [Equalizer1], [Equalizer2], etc. Each definition will be displayed in the private chat window under the Appearance menu when you right click in the window.

Name = (Topic name)

This is the human readable Appearance name. If not specified it will default to the topic name.

ColorBackground = (System setting)

Specifies the color to use when drawing the base background of the window.

ColorFreq = (System setting)

Specifies the color to use when drawing the frequency text above the sliders.

ColorScale = (System setting)

Specifies the color to use when drawing the scale text to the left of the sliders.

ColorValue = (System setting)

Specifies the color to use when drawing the current values text below the sliders.

ChkName = ChkMain

Specifies the topic name to use when reading theme settings for any check boxes in this Window.

BtnName = BtnMain

Specifies the topic name to use when reading theme settings for any push buttons in this window.

SliderName = Slider

Specifies the topic name to use when reading theme settings for any sliders in this window.

SliderVertMin = 156

Specifies the vertical minimum size in pixels for the sliders. The actual size of the slider can be larger depending on the graphical requirements of the slider control. The value must be in the range of 100 to 500.

SliderHorzSpacing = 3

Specifies the pixel spacing between each slider control. The value must be in the range of 0 to 100.

SliderHorzTest = -2

If theme designer mode is enabled this option creates a horizontal slider next to the close button. This slider serves no purpose and is meant for the designer to test a horizontal version of the slider since both types would exist in the same slider folder.

Possible values are:

-2 Disabled
-1 Create a -20 to 0 slider. (Negative only)

0 Create a -20 to 20 slider. (Negative and Positive, split slider)
1 Create a 0 to 20 slider. (Positive only)

UseBorder = EqualizerBorder

Specifies the topic name to use for this windows border. Ex: UseBorder=EqualizerBorder1. Implies you have a topic named [EqualizerBorder1].

[EqualizerBorder]

This topic describes the keys used to control how the Equalizer window border is displayed. See the documentation above for the [MainBorder] topic as the keys and values used are the same.

[SbcMain]

This topic describes the keys and values that control how a Scroll Bar is displayed. SBC stands for Scroll Bar Components.

Note: When using indents (normal or relative) you should define a new topic name and point the intended scrollbar at the new name. Indents are unique to a specific window and if you use them in the Main window and use the SbcMain as the default topic then those will be used in other windows if they attempt to use the same SbcMain topic as well. If using indents on Main window or Channel windows you should use the Keys “SbcNameMain” and “SbcNameChannel” to point to your indented version of the SBC.

Each of the four Indent keys has a variant that has a “Rel” attached to the end of it. This means the indent is relative to the inside edge of the border rather than the edge of the window the scroll bar resides in. This way as the window configuration changes the indent remains semi-static, such as when the main window is placed in extended mode where the User Name, Server and Bindings pulldowns are not visible, or the title bar is hidden.

Solid = 0

This key tells the program if the scroll bars (vertical and horizontal) are solid or not. Solid means the entire area covered by the scroll bar has nothing underneath it. Think of Solid has being the same as a standard Windows scroll bar, rectangular in shape and not transparent. Setting Solid = 0 tells the program that the scroll bars are not rectangular blocks but instead are irregular shapes that will expose

more information around them. This allows text and graphics to flow beneath and around the scroll bar exposing more content and/or background colors and images giving the window a more artistic look.

CornerOff = 0

When set to 1 turns off the display of the scroll bar corner. Should only be used when an appropriate HorzIndentRight and VertIndentBottom have been set. Otherwise the two scroll bars will overlap each other.

HorzIndentLeft = 0

Indents the left side of the horizontal scroll bar by X pixels.

HorzIndentRight = 0

Indents the right side of the horizontal scroll bar by X pixels.

VertIndentTop = 0

Indents the top of the vertical scroll bar by Y pixels.

VertIndentBottom = 0

Indents the bottom of the vertical scroll bar by Y pixels.

HorzIndentLeftRel = 0

Similar to HorzIndentLeft except it is relative to the inside edge of the border instead of the tree window.

HorzIndentRightRel = 0

Similar to HorzIndentRight except it is relative to the inside edge of the border instead of the tree window.

VertIndentTopRel = 0

Similar to VertIndentTop except it is relative to the inside edge of the border instead of the tree window.

VertIndentBottomRel = 0

Similar to VertIndentBottom except it is relative to the inside edge of the border instead of the tree window.

Folder = SbcMain

This key tells the program to look in a sub-folder of the current theme for graphic files to be used when displaying this specific Scroll Bar. For example, if you have large scroll bars for the main

window and small scroll bars for channel windows it is easier to use the default file names for each variation and place the individual files in different folders. If no folder is specified for this scroll bar the default files are assumed to be in the current theme folder.

Examples:

Folder = SbcMain

The default value will use the SbcMain folder in the Default theme if your theme does not have one.

The following keys can be used to override the default file name for individual scroll bar components. However, we recommend that you not change the names if you are providing different scroll bars for different windows. Instead, leave the names as they are and use the "Folder" key as documented above and use the same file names but different folders. If the following files are not found in the current theme folder then the program will use the files from the Default theme folder.

Vertical scroll bar keys

VertArrowUp = sbc-vert-uparrow.png
VertArrowDown = sbc-vert-downarrow.png
VertBackground = sbc-vert-background.png
VertButtonTop = sbc-vert-top.png
VertButtonBottom = sbc-vert-bottom.png
VertButtonMiddle = sbc-vert-middle.png
VertButtonOverlay = sbc-vert-middleoverlay.png
VertUpOverlay = sbc-vert-upoverlay.png (Optional)
VertDownOverlay = sbc-vert-downoverlay.png (Optional)

Horizontal scroll bar keys

HorzArrowLeft = sbc-horz-leftarrow.png
HorzArrowRight = sbc-horz-rightarrow.png
HorzBackground = sbc-horz-background.png
HorzButtonLeft = sbc-horz-left.png
HorzButtonRight = sbc-horz-right.png
HorzButtonMiddle = sbc-horz-middle.png
HorzButtonOverlay = sbc-horz-middleoverlay.png
HorzLeftOverlay = sbc-horz-leftoverlay.png (Optional)
HorzRightOverlay = sbc-horz-rightoverlay.png (Optional)

Corner keys

Corner = sbc-corner.png

[BtnMain]

This topic describes the keys and values that control how a push button is displayed.

Manual = 1

When set to non-0 instructs the program to manually draw the button border. When set to 0 the Folder option is used and the 9 part button image components are used instead. Please note that manual mode is much faster especially on older machines and that using the 9 part image buttons can be very slow depending on the system.

Folder = BtnMain

Specifies the folder name to use when reading “btn-filename.png” images.

ButtonProp = (Default not used)

NoPressHover = Disables the hover state when the button is pressed.
StateImages = Forces all states to use unique single image. No overlays.

ColorBackground = $(DEFBLACK)

Specifies the background color to use when drawing a button.

ColorBackgroundGrad = (Color Not used)

Used only in Manual mode.

ColorPressed = 128,0,0

Specifies the color to display when the user is pressing the button.

ColorPressedGrad = (Color Not Used)

Used only in Manual mode.

ColorCheck = 64,64,64

Specifies the color to display when the button is in a checked state.

ColorCheckGrad = (Color Not Used)

Used only in Manual mode.

ColorFGEnabled = $(DEFBLACK)

Specifies the foreground color to use when drawing text and the button is enabled.

ColorFGDisabled = 127,127,127

Specifies the foreground color to use when drawing text and the button is disabled.

ColorFGPressed = $(DEFWHITE)

Specifies the foreground color to use when drawing text and the button is pressed.

ColorFGChecked = $(DEFWHITE)

Specifies the foreground color to use when drawing text and the button is checked.

ColorHover = $(DEFHOVER)

Specifies the color to use when drawing when drawing the user hovers over it with the mouse.

FontName = (System setting)

Deprecated. Will not be available in future version. Theme's should never specify a font due to language portability.

PtSize = 9

Deprecated. Will not be available in future version. User will have control over this feature eventually.

CheckBox = 0

When set to 1 makes a push button act like a check box. Otherwise it behaves like a normal push button.

ColorBorder = 172, 172, 172

Used only in Manual mode.

Multiple colors can be used for individual lines that form the border. You must first provide a ColorBorder = value. You can then provide additional lines with ColorBorder1 = value. ColorBorder2 = value, and so on. Numbers must be in sequence.

NoText = 0

Used only in Check boxes.

Possible values:

0 = Button text will be drawn.
1 = No button text is used.

[ChkMain]

This topic describes the keys and values that control how a check box is displayed. It is identical to [BtnMain] with the following differences.

Manual = 0

When set to non-0 instructs the program to manually draw the button border. When set to 0 the Folder option is used and the 9 part button image components are used instead. Please note that manual mode is much faster especially on older machines and that using the 9 part image buttons can be very slow depending on the system.

Folder = ChkMain

Specifies the folder to use when reading images such as 9 part button. Applies only when Manual = 0.

CheckBox = 1

Specifies that this button is in fact a check box. Setting to 0 will make it act like a push button.

NoText = 0

When set to 1 the check box relies on the graphic to provide a visual representation of what the check box is used for and will not display the text normally associated with the check box itself. However, a check box will always display fly over help in case the user does not understand the meaning of the graphic.

[ChkMuteSnd]

This topic describes the keys and values that control how the Mute Sound button in the Main window Mute bar will be displayed. It is identical to [BtnMain] with the following differences. It is designed so that a theme designer can create a ChkMuteSnd folder in their theme and replace the images as they see fit. The topic need only be provided to override default behavior.

Manual = 0

When set to non-0 instructs the program to manually draw the button border. When set to 0 the Folder option is used and the 9 part button image components are used instead. Please note that manual mode is much faster especially on older machines and that using the 9 part image buttons can be very slow depending on the system.

Folder = ChkMuteSnd

Specifies the folder to use when reading images such as 9 part button. Applies only when Manual = 0.

CheckBox = 1

Specifies that this button is in fact a check box. Setting to 0 will make it act like a push button.

NoText = 1

When set to 1 the check box relies on the graphic to provide a visual representation of what the check box is used for and will not display the text normally associated with the check box itself. However, a check box will always display fly over help in case the user does not understand the meaning of the graphic.

[ChkMuteMic]

This topic describes the keys and values that control how the Mute Mic button in the Main window Mute bar will be displayed. It is identical to [BtnMain] with the following differences. It is designed so that a theme designer can create a ChkMuteMic folder in their theme and replace the images as they see fit. The topic need only be provided to override default behavior.

Manual = 0

When set to non-0 instructs the program to manually draw the button border. When set to 0 the Folder option is used and the 9 part button image components are used instead. Please note that manual mode is much faster especially on older machines and that using the 9 part image buttons can be very slow depending on the system.

Folder = ChkMuteMic

Specifies the folder to use when reading images such as 9 part button. Applies only when Manual = 0.

CheckBox = 1

Specifies that this button is in fact a check box. Setting to 0 will make it act like a push button.

NoText = 1

When set to 1 the check box relies on the graphic to provide a visual representation of what the check box is used for and will not display the text normally associated with the check box itself. However, a check box will always display fly over help in case the user does not understand the meaning of the graphic.

[ChkMsg]

This topic describes the keys and values that control how the MSG button in the Main windows Mute bar will be displayed. It is identical to [BtnMain] with the following differences. It is designed so that a theme designer can create a ChkMsg folder in their theme and replace the images as they see fit. The topic need only be provided to override default behavior.

Manual = 0

When set to non-0 instructs the program to manually draw the button border. When set to 0 the Folder option is used and the 9 part button image components are used instead. Please note that manual mode is much faster especially on older machines and that using the 9 part image buttons can be very slow depending on the system.

Folder = ChkMsg

Specifies the folder to use when reading images such as 9 part button. Applies only when Manual = 0.

CheckBox = 1

Specifies that this button is in fact a check box. Setting to 0 will make it act like a push button.

NoText = 1

When set to 1 the check box relies on the graphic to provide a visual representation of what the check box is used for and will not display the text normally associated with the check box itself. However, a check box will always display fly over help in case the user does not understand the meaning of the graphic.

[ChkPTT]

This topic describes the keys and values that control how the PTT button in the Main windows Mute bar will be displayed. It is identical to [BtnMain] with the following differences. It is designed so

that a theme designer can create a ChkPTT folder in their theme and replace the images as they see fit. The topic need only be provided to override default behavior.

Manual = 0

When set to non-0 instructs the program to manually draw the button border. When set to 0 the Folder option is used and the 9 part button image components are used instead. Please note that manual mode is much faster especially on older machines and that using the 9 part image buttons can be very slow depending on the system.

Folder = ChkPTT

Specifies the folder to use when reading images such as 9 part button. Applies only when Manual = 0.

CheckBox = 1

Specifies that this button is in fact a check box. Setting to 0 will make it act like a push button.

NoText = 1

When set to 1 the check box relies on the graphic to provide a visual representation of what the check box is used for and will not display the text normally associated with the check box itself. However, a check box will always display fly over help in case the user does not understand the meaning of the graphic.

[CmbMain]

This topic describes the keys and values that control how a Combo box / Pull down is displayed. See the [BtnMain] topic as the keys and values are the same.

[Slider]

This topic describes the keys and values that control how a Slider is displayed.

Logging = 0

When set to 1 this option will enable logging of all slider graphic files that could not be found. By default the slider will decide how it's drawn by what files are available or missing, such as when a (Direction)-fixed file is found it will draw in fixed size mode, whereas if the fixed files are missing it uses the scalable (Direction)-background. It is valid only when theme designer mode is enabled.

TryDef = 1

When set to 1 this option allows the slider to search the [Slider] folder in the default theme for each graphic option. Since the slider draws according to which files are available in the slider folder this can be used to disable searching the Default theme Slider folder by setting the value to 0.

Folder = Slider

This key tells the program to look in a sub-folder of the current theme for graphic files to be used when displaying this specific Slider. If no folder is specified for this scroll bar the default files are assumed to be in the current theme folder.

Keep in mind that if you do not override the default folder and then create custom graphics and place them in your own copy of the Slider folder then all other windows will default to this slider unless otherwise specified.

Examples:

Folder = SliderEqualizer

The default value will use the Slider folder in the Default theme if your theme does not have one.

ColorRangeUp = (Default not used)

This option allows for a color to be drawn between the 0 position and the current positive (Up or Right) position of the slider button. Useful for adding color without creating fixed graphics.

ColorRangeDown = (Default not used)

This option allows for a color to be drawn between the 0 position and the current negative (Down / Left) position of the slider button. Useful for adding color without creating fixed graphics.

ColorFocus = $(DEFBLACK)

This option changes the color of the hatched box drawn around the slider when it has focus.

Image files

The following keys can be used to override the default file name for individual slider components. However, we recommend that you not change the names if you are providing different sliders for different windows. Instead, leave the names as they are and use the “Folder” key as documented above and use the same file names but different folders.

All of these files are optional except for the button parts. The (Direction)-idle.png, (Direction)-hover.png and (Direction)-disabled.png images must exist for the specified slider. These are used for drawing the slider button. Depending on the window in which the slider appears, a slider without arrows

or a background shaft may be desired and appropriate. The program leaves this up to the theme designer.

Sliders are similar to scroll bars in that the (Direction)-background.png image that displays the shaft part of the slider is self-replicating and permits resizing of the sliders. However, if you provide actual files in your folder such as the vert-fixedbacktop.png and vert-fixedbackbottom.png this will instruct the program to make the sliders the exact size of those graphics combined. The same applies to the horizontal equivalents. This makes for more interesting looking sliders and it also permits the optional vert-fixedforetop.png and vert-fixedforebottom.png which act like overlays but for the shaft part of the slider. Some themes will use this functionality for windows like the Equalizer to make them more pleasing to the eye.

The arrows and optional arrow overlays are not usually used on sliders but are provided as optional components should the theme designer choose to use them.

Vertical scroll bar keys

// Button

VertIdle = vert-idle.png
VertHover = vert-hover.png
VertDisabled = vert-disabled.png

// Repeating background

VertBackground = vert-background.png

// Arrows and arrow overlays

VertArrowUp = vert-uparrow.png
VertArrowDown = vert-downarrow.png

VertUpOverlay = vert-upoverlay.png
VertDownOverlay = vert-downoverlay.png

// Fixed size

VertFixedBackTop = vert-fixedbacktop.png
VertFixedBackBottom = vert-fixedbackbottom.png

VertFixedForeTop = vert-fixedforetop.png
VertFixedForeBottom = vert-fixedforebottom.png

Horizontal scroll bar keys

// Button

HorzIdle = horz-idle.png
HorzHover = horz-hover.png
HorzDisabled = horz-disabled.png

// Repeating background

HorzBackground = horz-background.png

// Arrows and arrow overlays

HorzArrowLeft = horz-leftarrow.png
HorzArrowRight = horz-rightarrow.png

HorzLeftOverlay = horz-leftoverlay.png
HorzRightOverlay = horz-rightoverlay.png

// Fixed size

HorzFixedBackLeft = horz-fixedbackleft.png
HorzFixedBackRight = horz-fixedbackright.png

HorzFixedForeLeft = horz-fixedforeleft.png
HorzFixedForeRight = horz-fixedforeright.png

Sliders are displayed in three different formats:

1) 0 thru N indicating the values are all positive.
2) –N thru 0 indicating the values are all negative.
3) –N thru N indicating a split divider that is negative and positive. See Equalizer for example.

If your slider is going to use the optional “Fixed” graphic files for the slider background then the following rules must be followed:

1) Positive sliders must include the Right/Bottom fixed graphic files.
2) Negative sliders must include the Left/Top fixed graphic files.
3) Split sliders (–N to N) must include Right/Bottom/Left/Top fixed graphic files.

Control.ini

A control.ini file can exist in a folder in order to override the default behavior of Chk (Check Box) or Btn (Button). For example, the default theme has a control.ini file in each of the following folders in order to prevent the text on the check box from being display alongside the graphic. ChkMuteSnd, ChkMuteMic, ChkMsg and ChkPTT.

This can be useful for anyone who creates assets that could be drag and dropped into a folder for other theme designers to use. However, it might still require color and transparency values to be changed by the designer himself.

The following is general information about controls and general usage.

9 Part buttons:

A 9 part button is a graphical way of making buttons with irregular shaped borders and graphics with no programmatic intervention. The name itself describes what the button is. Take a rectangle and divide it up into nine distinct parts which is 3 rows by 3 columns. The example graph below shows what the numbering format is for the files that comprise the parts.

1 2 2 2 3
4 5 5 5 6
4 5 5 5 6
7 8 8 8 9

If you download the theme "Transparent-Gradient" you will see an example of what a 9 part button looks like. After you install the theme be sure to enter theme designer mode and then Expand the theme. Once expanded you will find a folder in this theme called "BtnMain" which contains 36 different small images.

Each 9 part button has 4 different states and each state as 9 unique images. The states are:

btn-main
btn-hover
btn-checked
btn-pressed

Each of these four states has 9 unique images as denoted by the numbers 1 thru 9 as the trailing character of the PNG file name. The number tells the program which part of the button the image will be displayed, as noted on the graph table above.

The graph above has the numbers 2, 4, 5, 6 and 8 show up multiple times in the grid. This is because a 9 part button must be able to scale depending on the size of the text that will be displayed inside of it and are subject to replication. If the program were to display a single letter with a small enough font it is very possible that none of the numbers in the grid would be replicated. However, real world usage says this is not likely to happen. It is also important to account for different languages for standard buttons once that feature becomes available.

The program first calculates the size of the text to be displayed in the button and then decides on how many copies of the 2, 5 and 8 parts it will take to contain the text horizontally and then calculate how many copies of 4, 5 and 6 will be needed to display the text vertically.

The text of the button will always be contained in part 5 of the grid. At this time it does not draw into any of the other parts.

At the current time all of the images must be the same dimension.

State images:

Single image buttons, such as those in the title / tool / mute bars, must be able to display an indication that the users mouse is hovering over the button.

1. Normal mode

   This is done by drawing the base image of the button first and then the hover image will be drawn on top of it as an overlay.

2. State image mode

   When a button is defined as a state image the hover is displayed using a dedicated hover image that will be drawn instead of the main base image.

Theme directives:

Theme directives are commands in an INI file that start with the # (hash) character. The following is the current list of theme directives.

```
#include "Local-File.ini"
#include <Default-File.ini>
```

This directive has two forms. One uses quote marks to surround a file name, the other uses paired < and > characters. The quotes version implies that the file you are trying to include is in your theme folder. The < > version tells the program to include the file from the Default theme. At the time of this writing there are no default theme INI files you can include but the functionality was added should such files become available in the future.

This directive is similar to the C and C++ programming languages for including header files. The Ventrilo client uses them for moving large portions of the script out of the settings.ini file making them easier to read and work on. For example, if you have numerous Appearances for the Main and Channel windows it would be helpful to move them in to their respective INI files using the following in the settings.ini file.

```
#include "main.ini"
#include "channel.ini"
```

Where ever a #include is encountered the program will pause reading the current INI file and begin reading the file specified in the #include statement. After that file has been completed it will resume reading the current file where it left off.

The file to be included is always at the top of your theme folder next to the settings.ini file. You can move the included file into a sub-folder but you must spell out the sub-folder name.

Example: #include “subfolder/channel5.ini” where “subfolder” is a folder directly below your theme folder.

Note: This directive does not support string substitution from the [Macros] topic.

Note: Some older themes might have comments that start with the # and should not be interpreted as directives.

Properties explained:

Properties are configuration options appended to a key that alters its normal behavior. Some properties have a different meaning depending on what topic / key they are applied to.

Left
Top
Right
Bottom
Center

These items are used for specifying an edge that the key is relative to. Usually used in reference to the window edge that separates the content area from the border. The context that this property is used is important whether it be content or border or some other window object.

Align

Function is relative to the context in which it is used.

When used in the SCP it will force the top of the SCP to be relative to the titlebar of the window instead of the very top edge of the window, and on a theme that has a very tall top border this could be sizeable distance and not appealing to the eye.

When used on “mbi-corner*.png” images and the associated border is using a fixed repeat count, like Repeat 3, it will force the corner image to indent and meet the end of the repeated border effectively putting a cap on the end of the border. These repeated borders will dynamically shrink if the window size become smaller than the repeated border allows for.

Under

When used in the MainTree window it allows the image to be drawn underneath the scroll bars instead of being shifted up and/or left when the scroll bars are visible.

When used in borders it forces an ImgOverlay (mbo-) to be displayed underneath the main border image (mbi-) rather than on top of it which is the default behavior.

Clip

When used in the MainTree it was meant to help reduce flicker. Not sure it provides that much benefit any more in this context.

When used on border overlay images (mbo-) it forces the overlay to be clipped to the edge it might overflow. For example, if you are using a border overlay that is displayed on the top of the window but the overlay is larger than what the user has horizontally sized the window to be, this option will clip the overlay at the left and right edges of the content part of the window. In this scenario not clipping the overlay may lead to the overlay drawing past the vertical edge and not looking very nice.

NoClip

Not used at this time. Left over from older version of the software.

Tile

This option is used for repeatedly drawing the specified image vertically and horizontally to cover the entire area.

MatchBorder

This option is used for assuring that an internal background will line up perfectly with the border pattern creating a seamless image. Usually used in conjunction with the Tile property.

Stretch

This option tells the program to stretch the image to cover the required area. It has limited value as it will distort the image vertically or horizontally and in the process make the image very unpleasant to the eye. It may also impede system performance.

Aspect

This option tells the program to maintain the aspect ratio of the specified image while it is being stretched or shrunk, most likely due to the user changing the size of the window. Be aware that aspect modes can be very slow on older systems.

NoPressHover

This option tells a control such as a button that it should not draw the hover overlay when the button is being pressed by the user.

StateImages

This option is used primarily on buttons and check boxes in order to simplify how the control is displayed. Instead of combining pressed and hover states over a buttons normal draw state, this option tells the program to use separate images that have already been defined by the designer. At this time there are 5 states that can be drawn. Active / Inactive / Disabled / Hover / Pressed.

Erase

This option will force the program to erase the entire rectangle where the specified image will be drawn allowing the image to take its place. The erase will clear the background to be totally transparent giving the graphic (which would most likely have alpha channel data in it) the ability to leave sections untouched and still transparent. Useful for cutting out sections of borders or content area to further enhance an irregular border appearance.

Alpha #

This option allows the theme designer to apply a global alpha transparency value to the entirety of the specified image. The # must be a value between 0 and 255 where 0 is total transparent and 255 is totally opaque. If the provided image already has alpha channel information then this option will make those parts of the image even more transparent i.e. cumulative.

Repeat (#)

This option instructs the program to repeat a feature it is associated with. For example, it can be used with Main border images (mbi-) that reside outside of the content area. If you have an image that is 10x10 but the height of the window is 10x100 then this option will repeat the specified border image until it covers the entirety of the specified edge of the window. The (#) parentheses are for documentation purposes only and indicates that the repeat count is optional. If a count is not provided it means to repeat until completed. If a number is provided like “Repeat 3” it means the border image will be repeated only 3 times.

Ofs #1 #2

This option allows an image to be shifted left, right, up and down in increments of pixels. The parameters as specified as #1=X and #2=Y. The actual implementation of this property is dictated by how the image is being processed.

Note: The parameters used to be specified as #1x#2 and separated by an “x”. The old format still works but we strongly recommend using the new form of a space character.

HorzPara #1 #2 #3

This property creates a horizontal parallax for the image it is applied to. It is comprised of 3 parameters with each parameter separated by a spaces.

#1 specifies the minimum horizontal width of the content part of the window before any parallax calculation is applied. Must be between 0 and 10000. Defaults to 0.

#2 specifies the maximum horizontal parallax shift in pixels that can be applied. Must be between -10000 and 10000. If you specify a negative multiplier then any value you give here must also be negative. Defaults to 0.

#3 specifies a negative or positive multiplier. If negative the image will be shifted to the left, conversely the image is shifted to the right if positive. This value must be non-0 for a parallax to be applied. Defaults to 0.

Example: “HorzPara 300 -200 -10” instructs an image to be shifted to the left after the window is at least 300 pixels wide, with a maximum shift of -200 pixels to the left and a multiplier of -10 with the negative indicating the shift should be to the left.

VertPara #1 #2 #3

This property creates a vertical parallax for the image it is applied to. It is comprised of 3 parameters with each parameter separated by a spaces.

#1 specifies the minimum vertical height of the content part of the window before any parallax calculation is applied. Must be between 0 and 10000. Defaults to 0.

#2 specifies the maximum vertical parallax shift that can be applied. Must be between -10000 and 10000. If you specify a negative multiplier then any value you give here must also be negative. Defaults to 0.

#3 specifies a negative or positive multiplier. If negative the image will be shifted to the top, conversely the image is shifted to the bottom if positive. This value must be non-0 for a parallax to be applied. Defaults to 0.

Example: “VertPara 400 -300 -20” instructs an image to be shifted to the top after the window is at least 400 pixels tall, with a maximum shift of -300 pixels to the top and a multiplier of -20 with the negative indicating the shift should be to the top.

Hue #1 #2 #3

This property is used for applying a color modifier to an Img key. The intended purpose is for the theme designer to make a grayscale image once and placed in a common area of the theme so that other parts of the theme, such as a channel window or main window appearance, can access the image but display it using a different color. This is a very easy technique when you have different appearances that simply change the color. It also means that you do not need to create multiple versions of the same image for each individual color. As you make changes to the source image the color variants are automatic.

If any part of the image is something other than a grayscale the result for an effected pixel is to decrease a particular color component from the pixel.

The parameters are #1=Red, #2=Green and #3=Blue. Each of the parameters should be viewed as a percentage of the color where 0 = 0% and 255 = %100 and 128 = %50.

If the source pixel is 255,255,255 and the property is “Hue 0 128 255” the resultant pixel will be 0,128,255.

If the source pixel is 128,128,128 and the property is “Hue 0 128 255” the resultant pixel will be 0,64,128. 0% of 0 = 0, 50% of 128 = 64 and 100% of 128 = 128.

Specifying 255 for a color component will leave the source pixel untouched. Any value less than 255 will decrease the source pixel by a percentage.

The alpha channel component of each pixel will remain untouched.

Please note that technically this property can take 4 parameters. When specified as four parts the first parameter is an Alpha value, #2 becomes Red, #3 becomes Green and #4 becomes Blue. However, the ability to affect the Alpha part of a pixel has been disabled at the time of this writing.

Publish folder:

An official published theme designer must create a folder in the themes root called “publish” and using all lower case. This folder must contain the following items.

PNG files:

example1.png
example2.png
example3.png
example4.png

These example images are displayed when a person is browsing the list of available themes and they zoom in on a specific one.

1. Example should be no wider the 600 pixels.
2. Example should have transparency if the border does.

In Theme Designer mode use the “PrtScr” key to take a screenshot of the window. Transparency will be maintained this way. Set focus to the window you intend to take a screenshot of. Click the PrtScr key, wait 5 seconds and the image will be produced. The 5 second delay will give you a chance to change the focus for the window depending on if you want it to have focus or not since your graphics could be different between the two states. If you open the “Working Directory” for your development environment you should see a folder called “ScreenShots”.

img128.png
img256.png

These two images are displayed on the website when browsing the list of available themes.

The img128.png should be no larger than 128 x 128 pixels. It will be displayed on the website in thumbnail form. You should scale the image to match the vertical or horizontal 128 requirement while trying to maintain aspect ratio. But neither dimension shall be larger than 128 pixels.

The img256.png should be no larger than 256 x 256 pixels. It will be displayed on the website when a user zooms in on a specific theme for more details. You should scale the image to match the vertical or horizontal 256 requirement while trying to maintain aspect ratio. But neither dimension shall be larger than 256 pixels.

If your theme contains transparency then these files should also have transparency. See the section above on example images on how to take a screenshot and maintain transparency.

Description files:

descr.txt

This file should contain a brief one or two sentences describing the theme.

1. No line breaks between sentences.
2. No html or formatting information.
3. ANSI only. UNICODE not allowed.

descr.html

This file should contains information that relates to special operational instructions about how to use the theme as intended. While it is in html format the only thing that should be html should be font sizes, bold, underlining, paragraph indenting and such styles of html code. No links to websites or anything else will be accepted. Any other formatting that is included will produce an automatic rejection of your theme for publication.

GUID file:

guid.txt

This file contains two lines of text. The first line is a GUID (Globally Unique ID) to identify the theme. The second line contains the version number of the theme.

Example:

```
{2a60af85-7f99-4956-8b8f-d69ebf6dc361}
4.0.0.0
```

The first line is the GUID surrounded by braces `{` and `}`. To generate a new GUID use one of the following commands that best matches your operating system. If the tool does not automatically place the braces around the GUID then be sure to add them in and with no spaces.

Windows 8: Open a Windows Powershell. Type = `[guid]::newguid()`
Windows 10: Open a Windows Powershell. Type = `new-guid`
Apple OS X: Open the Terminal Application. Type = `uuidgen`

Once you generate a GUID for a theme never change it. You should also be careful when starting a new theme from an existing theme as you will be copying the GUID along with it. In these cases be sure to create a new GUID for the theme you are starting.

The second line is the version number of your theme. It is comprised of 4 parts separated by periods. The first three parts should specify the base version of the Ventrilo Client for your theme to work correctly. With each successive update to your theme you should bump the 4th part of the version. For example 4.0.0.0 becomes 4.0.0.1, then 4.0.0.2, etc.

This file must be in ANSI format, not UNICODE.