Audio Visualizations

Totally Hip Technologies Inc.

Written by Steve Israelson
Waveform

Introduction 17
Creating Waveforms in LiveStage 18
Waveform Settings Tab 18
  Source Audio 18
  Sample Rate 18
  Use native rate 19
  Channel 19
  Averaging 19
  Vertical Scaling 19
  Horizontal Scaling 19
Waveform Graph Tab 20
  Time Cursor 20
  Horiz Grid 20
  Vert Grid 20
  Label Size 20
  Show X axis 20
  Show Y axis 21
  Axis Background Color 21
  Axis Border Color 21
  Axis Ticks Color 21
  Axis Labels Color 21
  Waveform Foreground Color 21
  Waveform Background Color 21
  Time Cursor Color 21
  Grid Color 21
Waveform Notes Tab 22
  Display Mode 22
  Edit Mode 22
  Default Note Color 22
  Default Note Data Color 22
  Selected Note Color 22
Waveform Scripts Tab 23

Spectrum Analyzer

Introduction 27
Creating Spectrum Analyzers in LiveStage 28
Spectrum Analyzer Settings Tab 28
  Source Audio 28
  Sample Rate 29
  Use native rate 29
  Channel 29
  FFT Size 29
  Filter Window 29
  Overlap 29
  Brightness 30
  Contrast 30
  Average FFTs 30
Mode 30
Decay Rate 30
Fade Rate 31
Zero Point 31
Rotation 31
Scale 31
Translate 31
Fade Out 31
Show Only Peak 31
Background and Waveform 32

Spectrum Analyzer Images Tab 32
Image List 32
Image Preview 32
Name 32
Steps 32

Spectrum Analyzer Bands Tab 33
Band List 33
Background 33
Foreground 33
Min Freq 33
Max Freq 33
Scale 34

Notes
Introduction 37
Creating 38
Editing 38
Grouping 38
Viewing 39
Meta data 39
Usage 40

Wired Actions
Introduction 43
Targeting 43
Handlers 43
Component Version 43
Actions 44
Sound Processing 44
Display 49
Notes 54
Spectrum Analysis Settings 68
Other 71
Introduction
Introduction

Audio visualizations are new visual track types added to QuickTime via Totally Hip's new Spectrograph components. These visualizations allow the multimedia author to visually display audio in their movies. There are three visualization types: Spectrogram, Waveform, and Spectrum Analyzer. Any number of these tracks can be present in a movie and can be configured in a variety of ways.

Creating these tracks is as simple as selecting one of these tracks from LiveStage's New Track menu. Once created, you can then configure the track to appear as you like.

The Audio Visualizations components were developed by Totally Hip Technologies Inc. for the Cornell Lab of Ornithology - Macaulay Library & Bioacoustics Research Program. For more information about Totally Hip, please visit: <http://www.totallyhip.com>. For more information about the Lab, please visit: <http://www.birds.cornell.edu>.

Spectrogram

The Spectrogram track shows a frequency versus time view of the audio. The color of each pixel represents the energy level of the audio at that point in the graph. You can change the color scheme from the default grey scale to any scheme you want. You can also configure the axes, grid, and play head. The spectrogram will be calculated in real time and will scroll as your sound plays. You can also configure it to show as little or as much as you want of the source audio.

Waveform

The Waveform track shows an energy level versus time view of the audio. This is how most people have seen audio represented. You can change the color of the waveform as well as the color of its background. The waveform is rendered in real time and will scroll as your sound plays. You can also configure it to show as little or as much as you want. As you zoom in to more closely inspect the waveform, it will automatically alter the way it is drawn to show you the most information.

Spectrum Analyzer

The Spectrum Analyzer shows the energy level of various frequency bands at the current instant in time. This can be configured as a bar graph or as an iTunes® like visualization. You can see the most basic bar graph mode at the right. You can customize this bar graph by adding images for each bar and by doing so, create displays like those seen in iTunes. There is a third display mode that allows you to create a visualization that rotates, zooms and fades as the sound plays.
Wired Actions

Many new wired actions have been added that support the audio visualizations. These wired actions let you control all of the settings of the audio visualization tracks.

Using these wired actions you can create your own UI as a part of your wired movie to control the visualization. You can create buttons in your wired movie that preset many settings at once to create different themes for your visualization.

For scientific studies you can add buttons that allow the user to fine tune all the parameters that affect the visual output in order to more clearly see the information desired.

Notes

Annotations can be added to both the spectrogram and waveform visualizations. Notes highlight a portion of the sound in both time and frequency. These notes can be any color and have any meta data desired associated with them. Text captions can be positioned at a variety of places around the note.

The authoring of notes can be done interactively by the user or they can be done through XML. Notes also have event handlers, such as mouse click, that can have scripts added to them so that actions can be taken when a note is clicked on or moused over.
Introduction

Sound consists of vibrations of the air. These vibrations can be represented as a waveform, but this waveform is very difficult to use to decipher a sound. If we analyze the waveform into its frequency parts, we will have a spectrogram. Spectrograms make deciphering a sound much easier.

Frequency is merely the notion of an event that repeats in time. In the case of a sound waveform the repeating even is the rise and fall of the waveform. Frequency is measured in Hertz which can be abbreviated as Hz. If something is happening 100 times per second, then it has a frequency of 100 Hz. If something happens at 1000 times per second, then its frequency is 1000Hz, or 1 kHz.

The range of human hearing is from 20Hz to 20kHz. Most human speech is below 8kHz, while some animals like birds and bats can have very high frequencies approaching 100kHz.

A mathematical technique called Fourier analysis, when applied to a waveform reveals what frequencies are present at any moment in time of that sound. This analysis is very computationally expensive. Fortunately there was a very fast way to do this analysis discovered call the Fast Fourier Transform, or FFT for short. We will use FFT throughout this document to refer to this transformation. Plotting these results over time produces a spectrogram.

The spectrogram at the top of this page represents about 20 seconds of sound. The horizontal axis represents time and the vertical axis represents frequency. The brighter color represents more acoustic energy, so we can easily see that this sound has a lot of lower frequency sound in it until the end.

Spectrograms are usually displayed in a gray scale color scheme where the darker color represents more acoustic energy. Other color schemes exist and you can fully customize the color scheme you want to use in displaying your spectrograms as seen here.

Spectrograms can also be annotated using Notes. These notes will mark an area of interest in the spectrogram. The notes can also display textual information, such as an author’s name or a description.
Creating Spectrograms in LiveStage

When in LiveStage Professional, choose the menu item: Movie>Create Track>Spectrogram. This will create a new audio visualization track configured as a spectrogram. If you get an error, or get an error when compiling, make sure you have properly installed the Spectrogram component into the proper location for QuickTime plug-ins.

The spectrogram track is like any other visual track in QuickTime. You can enable it, position it, rotate it and apply various drawing modes to it. It is also a wired action savvy track in that it allows you to add QScript wired actions to the track.

When you first create a spectrogram track or double click on one in the stage or time line, you will be presented with a settings window where you can alter the settings for the spectrogram display.

Spectrogram Settings Tab

The settings tab is where you make the most important settings that affect the spectrogram the most.

The settings are broken into two groups, one for the basic sound processing settings, and the other for FFT (Fast Fourier Transform) settings.

There is also a summary display that summarizes your settings.

An itemized list of each setting will be presented here.

Source Audio

This popup allows you to choose where the audio information comes from to create the visualization. Only Sound tracks and Movie in a Movie tracks are shown. You can also choose None, in which case the first sound track found is used.

If you choose a Movie in a Movie track, then the first sound track found inside that movie will be used. There will be times when the movie in this track will still be downloading from the server. Sound data will be processed once it becomes available in this case.

Multiplexed audio/visual files like MPEG 1 and streaming audio files are not supported.

Sample Rate

You can enter a sample rate that you want the sound to be sampled at in Hz. If you enter a sample rate that is less than the native sample rate of the sound track, then QuickTime will automatically down sample the sound using its built-in high quality down sampler. You would normally use this setting when you know that
the sounds you are interested in are of a much lower frequency than the native sample rate of the track. The sample rate you enter here must be at least twice the highest frequency you are interested in.

**Use native rate**

You can check this box if you want the native sample rate of the sound track to be used. This is especially useful when using a Movie in a Movie track as the source audio since the sample rate of each sound track that gets loaded into it may not be known ahead of time.

**Channel**

You can extract audio samples from either the left or the right audio channel. Some sound tracks are not in stereo, and in this case the left channel is what you should select. If you select the right channel and there is no right channel, then the left channel will automatically be used.

**FFT Size**

You can choose from values 64 to 2048 samples. The smaller the number you choose increases the time accuracy of the FFT and the larger the number you choose increases the frequency accuracy of the FFT. This is because the FFT function returns a number of frequency results that is proportional to the FFT size. The number of audio samples used to create this data is also in proportion to the FFT size. So if more audio samples are used, a bigger portion of the sound and thus time is used to create the FFT results. That means that multiple short changes in the sound at that time get grouped together when doing the FFT processing and thus are no longer distinguishable as separate changes.

**Filter Window**

You can choose between none, Blackman, Hann, Gaussian 4, and Welch. The none filter passes the audio samples through to the FFT in their raw format. This introduces noise into the signal since the audio waveform gets chopped up at arbitrary points. These other filters dampen down the waveform at either end of its chopped off points to help eliminate this noise.

**Overlap**

The overlap determines how much each chunk of sound that is taken for analysis overlaps the previous chunk. An overlap of 0 percent means that each chunk is taken immediately after the previous chunk. An overlap of 50 percent would mean that the next chunk is taken starting half way through the previous chunk. Setting an overlap gives you a way to increase the time accuracy of the spectrogram without impacting the frequency accuracy. If the overlap is negative, then it becomes a gap. An overlap of negative one hundred percent would mean that one whole FFT of samples is skipped between each chunk. Doing this can help you show more visual information in the track at once.
Brightness

The brightness increases the signal level of the results of the FFT. So, if the FFT results range from 0 to 50, setting the brightness to 10 will alter the range to be from 10 to 60. Since we are using a 256 color clut to display the results, we would ideally want the output values to range from 0 to 255. You will likely need to tweak this setting for each audio track you process because of the varying sound levels from one audio track to another. The brightness setting is applied after the contrast setting. This allows you to fine tune the visual output of the FFT.

Contrast

The contrast level increases the spread between the smallest signal and the largest signal. If your FFT results are not making use of the full range of colors in your CLUT, then you should increase the contrast. So, if the FFT results range from 0 to 50, setting the contrast to 5.0 will cause the results to range from 0 to 250. The contrast setting is applied before the brightness setting.

Average FFTs

If your overlap is a very large negative percent, indicating a big gap, then you can choose to perform more than one FFT on the audio data in this gap and average the results together to make just one FFT. The number of FFTs that will be averaged is determined automatically for you based on the gap size. This results in more accurate FFT data at the expense of extra processing.

Clut

If you want to use a clut that is different from the standard grey scale clut, then drag in a picture here that is exactly 256 pixels wide. The first row of pixels will be scanned and the colors of those pixels will be extracted for your clut. A cleverly created clut can make it much easier to examine the FFT results and gather information from them.

Spectrogram Graph Tab

The graph tab is where you make the settings that affect the axis and other visual settings.

You can turn on the X axis which will show the time along the bottom of the track. Turning on the Y axis will show the frequency along the left edge of the track.

The time cursor will indicate where in the spectrogram the current time is. This can be handy when you are listening to the sound to see what you are hearing.

The scaling allows you to determine approximately what duration of the sound will be visible at one time in the track. If not all of the sound will fit in, then the
track will scroll as time goes by so that the current time is centered when possible.

**Time Cursor**

Chose from None, Solid, Dotted, and Dashed. A solid line is very visible, but also obscures the spectrogram. The dotted line obscures less of the spectrogram, but is sometimes hard to see. The dashed line is a compromise between the two. The time cursor indicates the current time in the sound track and is drawn from top to bottom over the track.

**Horiz Grid**

The horizontal grid can be set to None, Solid, Dots, and Intersection. A setting of none will turn the horizontal grid off. Any other setting will turn the grid on and grid lines will be drawn at the major frequency ticks as shown in the axis at the left. The intersection setting will not draw a grid unless the vertical grid is also set to intersection.

**Vert Grid**

The vertical grid can be set to None, Solid, Dots, and Intersection. A setting of none will turn the vertical grid off. Any other setting will turn the grid on and grid lines will be drawn at the major time ticks as shown in the axis at the bottom. The intersection setting will not draw a grid unless the horizontal grid is also set to intersection.

**Label Size**

This setting controls the font size used when labeling the major ticks on the axes. You can choose sizes from 6 to 14 points.

**Show X axis**

This setting lets you control the visibility of the X axis. The X axis shows time and is drawn along the bottom edge of the spectrogram. When it is off, the spectrogram is enlarged to take up its space.

**Show Y axis**

This setting lets you control the visibility of the Y axis. The Y axis shows frequency and is drawn along the left edge of the spectrogram. When it is off, the spectrogram is enlarged to take up the space.

**Scaling**

The scaling setting lets you control how much of the spectrogram is visible at one time in the track. Choosing “Native” will render such that one FFT result takes up a width of one pixel. Choosing “Fit” will alter your overlap setting and your X scale setting so that the entire spectrogram will fit in the track at once. Choosing custom allows you to specify the duration of how much of the sound to show at once.
Axis Background Color
This setting controls the color used to erase the background of the axes.

Axis Border Color
This setting controls the color used to draw the border or edge of the axes.

Axis Ticks Color
This setting controls the color used to draw the ticks on the axes.

Axis Labels Color
This setting controls the color used to render the text labels on the axes.

Misc Background Color
This setting controls the color used to render areas of the track that do not currently show the spectrogram. You will see this color whenever the spectrogram is too small to fit in the track.

Time Cursor Color
This setting controls the color of the time cursor.

Grid Color
This setting controls the color of the grid.

Spectrogram Notes Tab
The notes tab is where you make settings that control the appearance of the notes. Notes can be turned off completely, set to only display the highlighted area, or to also display the textual meta data attached to a note.

When the edit mode is set to allow the creation of notes, then the user can click and drag out a selection directly in the spectrogram track. To convert that selection into a note requires some QScript, probably triggered by a wired button labeled “Make New Note”.

You can also attach a database of notes to the sound track by putting all the note data into XML format and attaching the XML to the sound track. Finally, you can download your notes database from a server and use QScript to attach the downloaded XML to the spectrogram track, either adding to or replacing the notes there.
Display Mode

This popup controls the display of notes in the spectrogram track. You can set this to “None” to display no notes, to “Notes only” to display just the notes, or to “Notes and meta data” to display the notes and the attached textual meta data.

Edit Mode

This popup controls the edit mode. You can set this to “None” to prevent the user from dragging out a new selection, or to “Create notes” to allow the user to drag out new selections to create new notes from.

Default Note Color

This is the color that a note that does not belong to any group will be drawn in.

Default Note Data Color

This is the color that note data will be drawn in if it does not have its own custom color.

Selected Note Color

This is the color a note will be drawn in if it is selected.

Spectrogram Scripts Tab

The scripts tab is where you add QScript to the spectrogram track. There are the track level frame loaded, key pressed, and idle events, and there are the standard mouse events for interaction with notes.

A special set of event handlers has been added so that you can detect and handle situations when the CPU is not powerful enough to render the Spectrogram fast enough. These handlers are named "Running Slow" and "Running Normal". When the Spectrogram begins to run too slow, then the “Running Slow” handler is triggered. When the Spectrogram returns to normal, then the “Running Normal” handler is triggered.

You can use these two handlers to either alter the FFT settings to reduce the CPU load, or display a warning graphic to the user so they can be informed that the settings they have chosen are too CPU intensive.

You can also add custom event handlers.
Introduction

Sound consists of vibrations of the air. These vibrations are usually represented as a waveform. This makes it easy to distinguish when a sound stops and starts and is handy when editing sound clips.

The waveform indicates the sound pressure level in the air at a point in time. As this pressure changes the waveform goes up and down. Louder sounds will have higher peaks and valleys, and silence will be a flat line.

There are issues to overcome when displaying a waveform. When zoomed way in, the waveform can be drawn as a curve, like a sine wave. As you zoom out, you can show energy levels that show the minimum and maximum levels in the range of audio samples represented by one vertical line in the plot. As you get more audio samples you can optionally show small groups of these samples overlaid over each other to get a better idea on how the energy is distributed through the waveform.

Here you can see three examples of the same sound at various zoom levels. The first image is showing a tiny portion of the sound representing a small fraction of a second of the sound. The second image is showing more of the sound, about one quarter of a seconds worth. the last image shows even more of the sound and is also using the averaging feature to blend smaller groups of sound samples into one plot.
Creating Waveforms in LiveStage

When in LiveStage Professional, choose the menu item: Movie>Create Track>Waveform. This will create a new audio visualization track configured as a waveform. If you get an error, or get an error when compiling, make sure you have properly installed the Spectrogram component into the proper location for QuickTime plug-ins.

The waveform track is like any other visual track in QuickTime. You can enable it, position it, rotate it and apply various drawing modes to it. It is also a wired action savvy track in that it allows you to add QScript to the track.

When you first create a waveform track or double click on one in the stage or time line, you will be presented with a settings window where you can alter the settings for the waveform display.

Waveform Settings Tab

The settings tab is where you make the most important settings that affect the waveform the most.

The settings are broken into two groups, one for the basic sound processing settings, and the other for waveform rendering settings.

An itemized list of each setting will be presented here.

**Source Audio**

This popup allows you to choose where the audio information comes from to create the visualization. Only Sound tracks and Movie in a Movie tracks are shown. You can also choose None, in which case the first sound track found is used.

If you choose a Movie in a Movie track, then the first sound track found inside that movie will be used. There will be times when the movie in this track will still be downloading from the server. Sound data will be processed once it becomes available in this case.

Multiplexed audio/visual files like MPEG 1 and streaming audio files are not supported.

**Sample Rate**

You can enter a sample rate that you want the sound to be sampled at in Hz. If you enter a sample rate that is less than the native sample rate of the sound track, then QuickTime will automatically down sample the sound using its built-in high quality down sampler. You would normally use this setting when you know that the sounds you are interested in are of a much lower frequency than the native sample rate of the track. The sample rate you enter here must be at least twice the highest frequency you are interested in.
Use native rate
You can check this box if you want the native sample rate of the sound track to be used. This is especially useful when using a Movie in a Movie track as the source audio since the sample rate of each sound track that gets loaded into it may not be known ahead of time.

Channel
You can extract audio samples from either the left or the right audio channel. Some sound tracks are not in stereo, and in this case the left channel is what you should select. If you have chosen the right channel and that channel does not exist, then the left channel will be used.

Averaging
This setting controls the way sound samples are processed when there are many sound samples used to create one single vertical plot in the waveform. The default is to calculate the minimum and maximum levels in that group of audio samples and plot that. You can turn on a more accurate blended mode that will divide the audio samples in up to eight groups and will plot each of those groups separately, overlaid on top of each other. This allows you to more accurately visualize the energy distribution in that group of audio samples.

Vertical Scaling
This setting controls the height of the waveform. Use this to scale the wave form so that you can clearly see all the details. If your waveform visualization draws as a thin line or as a colored blob, then this setting is either too low, or too high. This setting is needed due to the fact that every sound file has different recording levels.

Horizontal Scaling
This setting controls the horizontal scaling. You can either fit the entire waveform into the track, or specify the duration of the sound to plot in the track. When all of the waveform does not fit in the track, then it will scroll as the sound plays.
Waveform Graph Tab

The graph tab is where you make the settings that affect the axis and other visual settings.

You can turn on the X axis which will show the time along the bottom of the track. Turning on the Y axis will show the frequency along the left edge of the track.

The time cursor will indicate where in the waveform the current time is. This can be handy when you are listening to the sound and viewing the waveform to see what you are hearing.

Time Cursor

Chose from None, Solid, Dotted, and Dashed. A solid line is very visible, but also obscures the waveform. The dotted line obscures less of the waveform, but is sometimes hard to see. The dashed line is a compromise between the two. The time cursor indicates the current time in the sound track and is drawn from top to bottom over the track.

Horiz Grid

The horizontal grid can be set to None, Solid, Dots, and Intersection. A setting of none will turn the horizontal grid off. Any other setting will turn the grid on and grid lines will be drawn at the major energy ticks as shown in the axis at the left. The intersection setting will not draw a grid unless the vertical grid is also set to intersection.

Vert Grid

The vertical grid can be set to None, Solid, Dots, and Intersection. A setting of none will turn the vertical grid off. Any other setting will turn the grid on and grid lines will be drawn at the major time ticks as shown in the axis at the bottom. The intersection setting will not draw a grid unless the horizontal grid is also set to intersection.

Label Size

This setting controls the font size used when labeling the major ticks on the axes. You can choose sizes from 6 to 14 points.

Show X axis

This setting lets you control the visibility of the X axis. The X axis shows time and is drawn along the bottom edge of the waveform. When it is off, the waveform is enlarged to take up its space.
Show Y axis
This setting lets you control the visibility of the Y axis. The Y axis shows energy and is drawn along the left edge of the waveform. When it is off, the waveform is enlarged to take up the space.

Axis Background Color
This setting controls the color used to erase the background of the axes.

Axis Border Color
This setting controls the color used to draw the border or edge of the axes.

Axis Ticks Color
This setting controls the color used to draw the ticks on the axes.

Axis Labels Color
This setting controls the color used to render the text labels on the axes.

Waveform Foreground Color
This setting controls the color used to render the waveform. When the waveform has the blended mode turned on, the actual color you see will be lighter.

Waveform Background Color
This setting controls the color used to render the background of the waveform.

Time Cursor Color
This setting controls the color of the time cursor.

Grid Color
This setting controls the color of the grid.
Waveform Notes Tab

The notes tab is where you make settings that control the appearance of the notes. Notes can be turned off completely, set to only display the highlighted area, or to also display the textual meta data attached to a note.

When the edit mode is set to allow the creation of notes, then the user can click and drag out a selection directly in the waveform track. To convert that selection into a note requires some QScript, probably triggered by a wired button labeled “Make New Note”.

You can also attach a database of notes to the sound track by putting all the note data into XML format and attaching the XML to the sound track. Finally, you can download your notes database from a server and use QScript to attach the downloaded XML to the waveform track, either adding to or replacing the notes there.

Display Mode

This popup controls the display of notes in the waveform track. You can set this to “None” to display no notes, to “Notes only” to display just the notes, or to “Notes and meta data” to display the notes and the attached textual meta data.

Edit Mode

This popup controls the edit mode. You can set this to “None” to prevent the user from dragging out a new selection, or to “Create notes” to allow the user to drag out new selections to create new notes from.

Default Note Color

This is the color that a note that does not belong to any group will be drawn in.

Default Note Data Color

This is the color that note data will be drawn in if it does not have its own custom color.

Selected Note Color

This is the color a note will be drawn in if it is selected.
Waveform Scripts Tab

The scripts tab is where you add QScript to the waveform track. There are the track level frame loaded, key pressed, and idle events, and there are the standard mouse events for interaction with notes.

You can also add your own custom event handlers.
Spectrum Analyzer
Introduction

The Spectrum Analyzer track allows you to create interesting non-technical visualizations. These visualizations are similar to those seen in most stereo equipment, and MP3 players, like iTunes. Other more complex visualizations are possible, like the full screen visualizations in iTunes.

The spectrum analyzer makes use of the FFT to convert the audio waveform from energy level vs time to frequency vs time. It then plots the resulting information in several interesting ways.

The simplest display is the power display. This display takes the FFT data and uses each result to calculate the length of a vertical line to draw. The lowest frequencies are to the left, and the highest frequencies are to the right. The display changes very rapidly as the sound plays. You can turn on fade mode which will slowly fade out the previous display of the lines resulting in a ghosting effect as shown here.

The next display mode is the bars mode. This mode can be set up to display a bar graph chart in the way that stereo equipment does. You configure it by first supplying a set of images for each bar, one for the background and one for the bar fully lit up. You then configure each bar, specifying its min and max frequency and the bar images to use. You can have as many bars as you want. The energy level for the frequency range you specify is then used to determine how much of the lit bar to reveal over the background bar.

Finally, there is the waveform mode. This mode renders the frequency information as a waveform. You can perform a transformation on the display before each new waveform is plotted. You can make the waveform spin and zoom in, or just make it slide across the screen.
Creating Spectrum Analyzers in LiveStage

When in LiveStage Professional, choose the menu item: Movie>Create Track>Spectrum Analyzer. This will create a new audio visualization track configured as a spectrum analyzer. If you get an error, or get an error when compiling, make sure you have properly installed the Spectrogram component into the proper location for QuickTime plug-ins.

The Spectrum Analyzer track is like any other visual track in QuickTime. You can enable it, position it, rotate it, and apply various drawing modes to it.

When you first create a Spectrum Analyzer track or double click on one in the stage or time line, you will be presented with a settings window where you can alter the settings for the Spectrum Analyzer display.

Spectrum Analyzer Settings Tab

The settings tab is where you make the most important settings that affect the Spectrum Analyzer the most.

The settings are broken into three groups, one for the basic sound processing settings, one for the FFT settings, and the other for Spectrum Analyzer rendering settings.

The first two groups operate in the same way as they do for the Spectrogram track.

An itemized list of each setting will be presented here.

Source Audio

This popup allows you to choose where the audio information comes from to create the visualization. Only Sound tracks and Movie in a Movie tracks are shown. You can also choose “None”, in which case the first sound track found is used.

If you choose a Movie in a Movie track, then the first sound track found inside that movie will be used. There will be times when the movie in this track will still be downloading from the server. Sound data will be processed once it becomes available in this case.

Multiplexed audio/visual files like MPEG 1 and streaming audio files are not supported.
Sample Rate
You can enter a sample rate that you want the sound to be sampled at in Hz. If you enter a sample rate that is less than the native sample rate of the sound track, then QuickTime will automatically down sample the sound using its built-in high quality down sampler. You would normally use this setting when you know that the sounds you are interested in are of a much lower frequency than the native sample rate of the track. The sample rate you enter here must be at least twice the highest frequency you are interested in.

Use native rate
You can check this box if you want the native sample rate of the sound track to be used. This is especially useful when using a Movie in a Movie track as the source audio since the sample rate of each sound track that gets loaded into it may not be known ahead of time.

Channel
You can extract audio samples from either the left or the right audio channel. Some sound tracks are not in stereo and in this case the left channel is what you should select. If you choose the right channel and there is no right channel available in the audio track, then the left channel is used instead.

FFT Size
You can choose from values 64 to 2048 samples. The smaller the number you choose increases the time accuracy of the FFT and the larger the number you choose increases the frequency accuracy of the FFT.

Filter Window
You can choose between none, Blackman, Hann, Gaussian 4, and Welch. The none filter passes the audio samples through to the FFT in their raw format. This introduces noise into the signal since the audio waveform gets chopped up at arbitrary points. These other filters dampen down the waveform at either end of its chopped off points to help eliminate this noise.

Overlap
The overlap determines how much each chunk of sound that is taken for analysis overlaps the previous chunk. An overlap of 0 percent means that each chunk is taken immediately after the previous chunk. An overlap of 50 percent would mean that the next chunk is taken starting half way through the previous chunk. Setting an overlap gives you a way to increase the time accuracy of the spectrogram without impacting the frequency accuracy. If the overlap is negative, then it becomes a gap. An overlap of negative one hundred percent would mean that one whole FFT of samples is skipped between each chunk. Doing this can help you show more visual information in the track at once.
**Brightness**

The brightness increases the signal level of the results of the FFT. So, if the FFT results range from 0 to 50, setting the brightness to 10 will alter the range to be from 10 to 60. Since we are using a 256 color CLUT to display the results, we would ideally want the output values to range from 0 to 255. You will likely need to tweak this setting for each audio track you process because of the varying sound levels from one audio track to another. The brightness setting is applied after the contrast setting. This allows you to fine tune the visual output of the FFT.

**Contrast**

The contrast level increases the spread between the smallest signal and the largest signal. If your FFT results are not making use of the full range of colors in your CLUT, then you should increase the contrast. So, if the FFT results range from 0 to 50, setting the contrast to 5.0 will cause the results to range from 0 to 250. The contrast setting is applied before the brightness setting.

**Average FFTs**

If your overlap is a very large negative percent, indicating a big gap, then you can choose to perform more than one FFT on the audio data in this gap and average the results together to make just one FFT. The number of FFTs that will be averaged is determined automatically for you based on the gap size.

**Mode**

This setting lets you control what mode the Spectrum Analyzer displays. There are three modes you can choose from: Power, Bars, and Waveform.

- **Power mode** will render a set of vertical lines, one for each frequency level returned from the FFT. This means that the FFT size and sample rate will control how many bars there are. The bars will be evenly spaced to fill the track.

- **Bars mode** will make use of the images and bands to render a bar chart with each bar representing one band as configured in the other tabs.

- **Waveform mode** will take the same information as the lines mode but will instead render it as a continuous wave. You can then apply transformations such as scaling and rotation to this wave to create interesting effects.

**Decay Rate**

This setting applies only to the Bars mode. This controls the rate that the level shown in each bar diminishes over time. A rate of 0 means the bars will always show the current energy level for that band, otherwise the energy level displayed in the band will diminish at the rate specified. This means that when the bar jumps up to display a sudden loud sound, it will more slowly
return to a lower level as the volume goes down. This allows you to more easily see sudden peaks in the sound levels.

**Fade Rate**

This setting applies to the Power and Waveform modes only. Instead of completely erasing the previous display before drawing the new one, the previous display will slowly fade into the background. This can have a more pleasing effect. A fade rate of about 10000 is usually a good starting point. The fade rate is enabled by the fade out setting.

**Zero Point**

This setting applies to the Waveform display only. This controls the vertical position that the waveform is drawn at. This is most useful when the transformation you apply requires the waveform to be at a certain position to be properly visible.

**Rotation**

This setting applies only to the Waveform display. This controls how much the previous display is rotated before the new display is rendered. The rotation angle can be specified with fractional amounts and the display will be rotated about the center point specified.

**Scale**

This setting applies only to the Waveform display. This controls how much the previous display is scaled before the new display is rendered. The scale amounts can be fractional and the scaling will be centered at the point specified. The scaling will be an enlargement if the scale factor is greater than 1.0 and a shrinkage if it is less than 1.0.

**Translate**

This setting applies only to the Waveform display. This controls how much the previous display is translated before the new display is rendered. Translation is a movement in the X and/or Y direction. If the pixel values entered are negative, then the translation is to the upper left, otherwise it is to the lower right.

**Fade Out**

This setting controls the fade out mode. When on, the previous display fades out slowly as the new display is drawn. The fade rate is controlled by the fade rate setting.

**Show Only Peak**

This setting applies to the Bars display only. When on, only the top bar is drawn in the ON position. When off, all bars lower than the top most ON bar are also drawn.
Background and Waveform

These settings control the background and foreground colors for the display.

Spectrum Analyzer Images Tab

The images tab is where you add the images for use in the Bars mode display. The images you put here can be referred to in the Bands tab and can be applied to one or more bands.

The images you add here will represent one vertical bar in the bar chart. Each image needs to be created in pairs, one image with all the lights or bars fully lit, and one with the lights unlit. When drawn, the proper mix of the lit and unlit images will be chosen. The lit and unlit images must also be the same dimensions.

Image List

This list displays all the images you have added showing the image name and its index in the list. You can select images from this list in order to view the image and edit its settings. You can also delete the selected image by pressing delete.

Image Preview

The preview area will show the selected image. You will also be able to see the file name of the actual image file that this image comes from. Below the preview you can see the dimensions of the image.

Name

This setting displays the name for the image. You can enter any name you desire for the image. These names will be shown in the Bands tab when you choose the images for each band. It is best to clearly name the lit and unlit images so you can correctly set up each band in the next tab.

Steps

This setting allows you to specify the individual lights or bars in your image. If you specify no steps, then when your image is used to render a bar, it will get cut at the nearest pixel to represent the energy level in the bar. If you do specify steps, then those steps indicate vertical pixel positions that the image can be cut at. This is useful when your image consists of individual sets of bars or lights, like the image shown above.
Spectrum Analyzer Bands Tab

The bands tab is where you create each bar in the bar chart. These bars are called bands because they specify a certain frequency range or band that they will represent.

You can create as many bands as you want. Each band has a minimum and maximum frequency it represents. These frequency ranges can overlap other bands.

Each band can also have a scale applied to the energy data that is used to render the band. This is useful because the higher frequency sounds often have less energy and the bars towards the right of the display tend to never go all the way to their maximum level. You can use this scale to adjust the level of each band.

Band List

This list shows you all the bands you currently have. The list will always be sorted by the average frequency the band represents. Each item in the list will display its frequency range. You can delete a band by selecting it and pressing delete. You can create a new band by pressing the new band button below the list.

Background

This popup menu lets you choose one of the images from your images list to represent the background of this bar, or the bar in its unlit state.

Foreground

This popup menu lets you choose one of the images from your images list to represent the foreground of this bar, or the bar in its lit state.

Min Freq

This setting controls the minimum frequency that this band represents. When the results of the FFT are gathered, any that are greater than this frequency and less than the maximum frequency will be added to the energy level of this bar.

Max Freq

This setting controls the maximum frequency that this band represents. When the results of the FFT are gathered, any that are greater than this frequency and less than the maximum frequency will be added to the energy level of this bar.
Scale

This setting controls the scaling for this band. Normally this is set to 100 percent, but if in your experiments you find that this band is consistently too strong or too weak, then adjust this scaling.
Notes
Introduction

When studying a spectrogram, you can usually pick out certain patterns that you need to remark on. One way to do this is to print out your spectrogram, and then use a pencil. That method does not allow you to do much, other than write notes. The spectrogram track allows you to do a similar thing - add notes to the visualization.

Let's say you had a microphone set up to record sounds from your bird feeder in your backyard. When you later look at the spectrogram, you find that there are a lot of interesting sounds or a variety of things. You can use notes to highlight each separate sound and add information describing what that sound was made by. Maybe you have video along with your sound and you can see that some sounds were made by certain birds, while others were made by some squirrels. You may also study this recording over the course of several days, so you would also want to add in the date and time that you deciphered each sound. If you have more than one person working on this, you would also want to add information as to who was authoring the note. You could even add in a url for a website that has more information. You can do all this with the notes system that is a part of the audio visualizations.

Annotations (notes) can be added to both the spectrogram and waveform visualizations. Notes highlight a portion of the sound in both time and frequency. These notes can be any color and have any textual meta data desired associated with them. The meta data can be used as text captions that can be positioned at a variety of places around the note.

The authoring of notes can be done interactively by the user or they can be done through XML. Notes also have event handlers, such as mouse click, that can have scripts added to them so that actions can be taken when a note is clicked on or moused over.
Creating

The easiest way to create notes is to set the spectrogram track edit mode to “Create Notes”. Then the user can simply click and drag out a selection. Saving that selection as a note can only be done by triggering a wired action: \texttt{QSSp\_SaveSelectionAsNote}. You can trigger this wired action by having a button titled “New Note” that executes this wired action when clicked. When you call this wired action, you pass in a variable that will have the new note’s ID stored in it if saving the note was successful, otherwise it will be set to zero. Once you have this ID, you can then configure the note and add meta data to it using other wired actions.

If you have certain meta data that would be common to all new notes, it would be best to gather this data, possibly by allowing the user to enter it into an edit field, and then apply it to each new note created. You could do this for the author’s name, and the date and time. You would probably also want to have an edit field where the user can add a description for each note.

If you close the movie after creating notes, then all the notes will be lost. To save the notes you must export them to XML and then communicate this XML to a server where you can save the notes. You could also save the XML to a file and attach that xml to the audio track using LiveStage. Then, whenever the spectrogram opens this audio track, it will check to see if an XML file is attached to the track, and if so, will load the notes from it.

Editing

The user may want to fine tune the notes, once they have been added. There are several wired actions that allow you to nudge the four edges of the note. You could add these actions to a set of 4 nudge buttons that the user could press to fine tune the exact placement and dimensions of the note.

You can also extract the time and frequency range that the note covers and allow the user to edit those values directly. Once edited, you can re-assign those values to the note.

Grouping

A note can also belong to exactly one group. A group is simply a way to categorize a set of notes. You can create new groups and assign meta data to them. Each group can be configured to have a different color. These colors will be used to render the notes that belong to that group.

One use of groups is to have a group for each author. In this case you would want the meta data to have the author’s name. You could also use groups to represent different sound sources. In this case the meta data would describe the sound source - maybe a bird’s species.

To create a new group, use the \texttt{QSSp\_MakeNewGroup} action. You pass in a variable that will be set to the ID of the new group object. Once you have this variable, you can then configure the group. When you create a new note, you can assign it to a group using the \texttt{QSSp\_SetNoteGroup} action.
Viewing

Once you have a database of notes built up, you will want to be able to show it to the user and allow the user to sift through it. You can iterate through the notes using the `QSSp_GetNextNoteID` action. Once you have the next note, you can scroll so that it is visible using the `QSSp_ScrollToShowNote` action, and select it using the `QSSp_SelectNote` action. If the note is small, you can also use the `QSSp_ScaleToFitNote` action to change the spectrogram settings such that the note fills the track.

Meta data

The notes and groups can have textual meta data added to them. Each piece of meta data is identified by a unique key. Each key can only be used once per note or group. Once you add meta data with a key, any time you add more meta data with the same key, it will replace the old meta data.

So, for example, let’s say you use the key 1000 to represent the author. For each new note you make you would add the string for the author’s name “Fred” using the key 1000. Later, when you are examining a note and you want to know the author’s name, you would look at the string with the ID of 1000.

If you want a note to be visible, then you would need to set its display position using the `QSSp_SetNoteDataDisplayPos` action. If the position is set to `QSSp_kDisplayPosHidden`, then the note is not shown, otherwise it is displayed in the position indicated by the setting. You can also add `QSSp_kDisplayPosMouseOver` to the position of the note so that it is only displayed when the mouse is over the note.

There is no limit to the number of meta data tags you can have attached to each note. The organization of the keys is completely up to you. It is however recommended that you use the keys already defined by QuickTime:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>‘name’</td>
</tr>
<tr>
<td>Author</td>
<td>‘©aut’</td>
</tr>
<tr>
<td>Comment</td>
<td>‘©cmt’</td>
</tr>
<tr>
<td>Creation Date</td>
<td>‘©day’</td>
</tr>
<tr>
<td>Description</td>
<td>‘©des’</td>
</tr>
<tr>
<td>Director</td>
<td>‘©dir’</td>
</tr>
<tr>
<td>Disclaimer</td>
<td>‘©dis’</td>
</tr>
<tr>
<td>Full Name</td>
<td>‘©nam’</td>
</tr>
<tr>
<td>Host Computer</td>
<td>‘©hst’</td>
</tr>
</tbody>
</table>
Usage

The only limit on how you use the meta data is your imagination. Here are some suggestions.

Always add a short and long description to the note. Then you can display the short description above the note and the long description in a smaller font below the note maybe in response to a mouse over event.

If your audio file has more than one instance of the same sound, then add a note that is not displayed that contains either the time of the next occurrence of the sound, or the ID of the next note attached to the next sound. You can then use a script attached to the mouse click handler for the note that extracts this data and uses it to jump to the next occurrence of this sound.

If there are online references to similar sounds, then add a note that is not displayed that contains the url to that web reference. then when the user clicks on the note, they can be take to the web page that describes this sound. This could work really well if your movie is embedded in a web page and the description page is set as a frame.

You can use notes as markers to jump around in your audio file. You can turn off the display of notes and just make a list of links for each note. Clicking in each link could cause that note to be centered. This would make a kind of chapter track for the visualization.

You could also use a meta tag to describe how this note is used. Some notes could be markers, some could be urls, and others could be links in time.
Wired Actions

```plaintext
MovieVars myID
Qssp_GeteventNote|ID(myID) // Get the note ID clicked on
if(myID > 0)
    QSSp_SelectNote(myID, FALSE) // Select the Note clicked ON
    TrackNamed("Spectrogram").executeEvent(39064)
    Tracknamed("Nudge").setEnabled(TRUE)
else
    QSSp_DeselectAllNotes // deselect all notes.
endif
```
Introduction

The following wired actions can all be used on any of the three types of audio visualization tracks. Some of the settings will not be applicable to some of the track types. Where possible, all of the settings available from within LiveStage can be retrieved and changed using wired actions.

Targeting

If you do not supply a target for these actions, then the first audio visualization track found in the movie will be used as the target. If you have more than one audio visualization track in your movie, then you should explicitly target the appropriate track using the standard track targeting mechanism of TrackNamed, TrackOfID, and TrackOfSize.

Handlers

You can add wired actions to the various event handlers associated with the visualization track in the scripts tab. These handlers work just like sprite handlers in a sprite track. You can get the ID of the note object that triggered the handler by calling **QSSp_GetEventNoteID** and passing in a variable that will be filled in with the ID of the note that caused the event. If the event was not triggered by the user interacting with a note, then the note ID will be 0. This can happen if the user clicks in the track and not in a note.

You can also create and call custom event handlers. These handlers are attached to the track so when targeting them you only need to target the track.

Component Version

You can detect the existence of the Audio Visualization component and its version number using the **ComponentVersion** QScript action. This action can be in a small placeholder movie on your web site that can redirect users to a download page if the component is not found. The returned version number has a major and minor component to it. The major portion of the number can be determined by dividing the version number returned by 65536 using the integer division operator **DIV**. The minor version can be determined by using the **REM** operator. In most cases just using the resulting version number is appropriate. You must pass in ‘mhlr’ for the component type and ‘Swan’ for the component sub type. You can pass in 0 for the manufacturer.

Example: debugstr(ComponentVersion(“mhlr”, “Swan”, 0))
Actions

Here is a list of all the wired actions supported in the visualization tracks.

Sound Processing

The following actions all apply to the control and monitoring of the sound processing for the audio visualization tracks.

QSSp_SetSourceTrack(inID)

Sets the track ID of the audio track to be used as a source for audio samples needed for sound processing. In most cases the track ID is the same as the track index. This can be the ID of a “Movie in a Movie” track in which case the first audio track found within the currently loaded movie will be used. When a new movie is loaded into the MIAM track, the first audio track found in it will be automatically chosen. If no valid audio track is found, or if the track specified is not a valid audio track, then the first valid audio track in the movie will be used.

You can change the source audio track at any time. The visualization track will automatically begin to sample new audio data from this track.

Parameters:
inID An expression that evaluates to an integer that specifies the track ID of the track to extract audio information from.

Used in: Spectrogram, Waveform, Spectrum Analyzer

QSSp_SetChannel(inChannel)
QSSp_GetChannel(result)

Sets and gets the channel to extract audio data from. Valid values are 0 = left, 1 = right or use these constants instead: QSSp_kLeftChannel, QSSp_kRightChannel. If you specify the right channel of a monaural sound track, then the left or mono channel will be used instead.

Parameters:
inChannel An expression that evaluates to an integer that specifies the audio channel to extract audio samples from.
result A variable used to store the result in.

Used in: Spectrogram, Waveform, Spectrum Analyzer

QSSp_SetFFTSize(inValue)
QSSp_GetFFTSize(result)

Sets and gets the power of two to use for the FFT window size. Valid values range from 4 - 12. Changing the FFT size can cause the spectrogram or spectrum analysis to show a bigger or smaller range of audio in the track. This setting has no effect on a spectrogram track configured to show a waveform. A smaller FFT size produces a spectrogram that displays more accuracy in the time domain and less accuracy in the frequency domain. A larger FFT size produces a spectrogram with more accuracy in the frequency domain but less ac-
accuracy in the time domain. What this means is that a larger FFT size will show lots of detail vertically in the spectrogram, but horizontally will show less detail.

Here is a list of the valid FFT sizes and the equivalent FFT window size in samples.

<table>
<thead>
<tr>
<th>FFT Size</th>
<th>Sample Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
</tr>
<tr>
<td>8</td>
<td>256</td>
</tr>
<tr>
<td>9</td>
<td>512</td>
</tr>
<tr>
<td>10</td>
<td>1024</td>
</tr>
<tr>
<td>11</td>
<td>2048</td>
</tr>
<tr>
<td>12</td>
<td>4096</td>
</tr>
</tbody>
</table>

**Parameters:**

- **inValue**: An expression that evaluates to an integer and will be constrained to a value from 4 - 12.
- **result**: A variable used to store the result in.

**Used in:** Spectrogram, Spectrum Analyzer

QSSp_SetStepSize(inValue)
QSSp_GetStepSize(result)

Sets and gets the amount of time to jump forward in the sound for each FFT sampling. This time is in 600ths of a second. Valid values range from 1 - 600 or more. This time value will be converted into a number of audio samples and will get rounded off to the nearest number of audio samples. Changing the step size can cause the spectrogram, waveform or spectrum analysis to show a bigger or smaller range of audio in the track. You can also use the frames per step version of these calls if you would like to set the step size based on the number of audio frames instead.

**Parameters:**

- **inValue**: An expression that evaluates to an integer number greater than 0.
- **result**: A variable used to store the result in. The actual value may not be the same value that was set due to rounding.

**Used in:** Spectrogram, Spectrum Analyzer, Waveform

Wired Actions 45
QSSp_SetFramesPerStep(inValue)
QSSp_GetFramesPerStep(result)

Sets and gets the number of samples to jump forward in the sound for each FFT sampling. This number must be greater than 0. Changing the step size can cause the spectrogram, waveform or spectrum analysis to show a bigger or smaller range of audio in the track. You can also use the step size version of these calls if you would like to set the step size based on time instead.

Parameters:
inValue An expression that evaluates to an integer number greater than 0.
result A variable used to store the result in.

Used in: Spectrogram, Spectrum Analyzer, Waveform

QSSp_SetScaleToFit(inValue)
QSSp_GetScaleToFit(variable)

Sets and gets the track scaling. You can specify how much time to represent in track display. This time value is in 600ths of a second. Showing more time in the spectrogram will cause the spectrogram to be shrunk horizontally to fit the amount of time you specify into the track display area. The visual display will also take longer to process as the amount of time displayed goes up. There are two special constants you can use: QSSp_kFitAll and QSSp_kFitNative. These will set the appropriate settings such that the entire sound track is displayed in the visualization. Using native will reset the settings so that the step size matched the FFT size.

This setting will internally adjust the step size such that the entire audio track can be process in one buffer. The X scale will be adjust so that this buffer can be rendered to just fit into the visualization track.

Parameters:
inValue An expression that evaluates to an integer number greater than 0, or one of the two constants.
result A variable used to store the result in.

Used in: Spectrogram, Spectrum Analyzer, Waveform

QSSp_SetWindowType(inType)
QSSp_GetWindowType(result)

Sets and gets the windowing function for the FFT. The windowing function allow you to control the various ways that noise is filtered out of the audio prior to performing the FFT. Valid values are:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>QSSp_kWindowNone</td>
<td>0</td>
</tr>
<tr>
<td>QSSp_kWindowBlackMan</td>
<td>1</td>
</tr>
<tr>
<td>QSSp_kWindowHann</td>
<td>2</td>
</tr>
<tr>
<td>QSSp_kWindowGaussian4</td>
<td>3</td>
</tr>
<tr>
<td>QSSp_kWindowWelch</td>
<td>4</td>
</tr>
</tbody>
</table>

Wired Actions 46
You can also pass in an expression that evaluates to one of these values.

**Parameters:**

inType An expression that evaluates to an integer number from 0 - 4.
result A variable used to store the result in.

**Used in:** Spectrogram, Spectrum Analyzer

QSSp_SetSampleRate(inRate)
QSSp_GetSampleRate(result)

Sets and gets the sample rate that the source audio will be re-sampled at. Valid ranges are from 1hz to 44100hz. Even though the resampling is very high quality, this can introduce noise into your sounds when they are processed. You should set the sample rate to at least twice the frequency of the highest frequency sound you are interested in. Changing the sample rate can cause the spectrogram, waveform or spectrum analysis to show a bigger or smaller range of audio in the track.

When working with a Spectrum Analyzer and music audio, you may want to set this to about 5000 to get a pleasing spectrum display.

**Parameters:**

inRate An expression that evaluates to an integer number from 1000 to 44100.
result A variable used to store the result in.

**Used in:** Spectrogram, Spectrum Analyzer, Waveform

QSSp_SetUseNativeSampleRate(inValue)
QSSp_GetUseNativeSampleRate(variable)

Sets and gets the native sample rate flag. Setting this to true will cause the sample rate setting to be set to the sample rate of the audio track. The sample rate will be changed when new audio tracks are load via a MIAM track too. This allows you to avoid any down sampling noise that may be added when the sample rate you choose is lower than the sample rate of the source audio.

**Parameters:**

inValue An expression that evaluates to true or false.
result A variable used to store the result in.

**Used in:** Spectrogram, Spectrum Analyzer, Waveform

QSSp_SetWaveFormAveraging(inValue)
QSSp_GetWaveFormAveraging(variable)

Sets and gets the waveform averaging flag. When the waveform is calculated, multiple audio samples are used to plot one vertical slice of the waveform. Normally, the smallest and largest audio samples are determined and used to plot this slice. When averaging is turned on, these audio samples are divided into groups and the smallest and largest value in each group is used to plot a set of overlapping lines for all the groups in this slice. This produces a more accurate display of where the most power is in that portion of the waveform.
This method is only used when the step size is large enough to produce enough audio samples per vertical slice of pixels in the track display.

**Parameters:**

- **inValue**
  - An expression that evaluates to true or false.

- **result**
  - A variable used to store the result in.

**Used in:** Waveform

**QSSp_SetFFTAveraging(inValue)**

**QSSp_GetFFTAveraging(variable)**

Sets and gets the FFT averaging flag. If averaging is turned on and the step size is big enough that more than one FFT window will fit into each step, then multiple FFTs will be done in that step and they will be averaged together. This produces more accurate results at the expense of processing time.

**Parameters:**

- **inValue**
  - An expression that evaluates to true or false.

- **result**
  - A variable used to store the result in.

**Used in:** Spectrogram, Spectrum Analyzer

**QSSp_SetBrightness(inValue)**

**QSSp_GetBrightness(result)**

Sets and gets the spectrograph brightness. Valid values range from 0 - 256. This does not actually set the physical brightness of the spectrogram display. The actual colors seen will depend on the current clut. This will cause values towards the end of the clut to be used more when rendering. If those colors are brighter, then increasing the brightness will indeed make the display appear brighter. If those colors are darker, then increasing the brightness will make the display look darker. In either case, the FFT values calculated will be amplified as the brightness goes up.

**Parameters:**

- **inValue**
  - An expression that evaluates to an integer from 0 - 256.

- **result**
  - A variable used to store the result in.

**Used in:** Spectrogram, Spectrum Analyzer

**QSSp_SetContrast(inValue)**

**QSSp_GetContrast(result)**

Sets and gets the spectrograph contrast. Valid ranges are from 0.1 - 1000. When used on a waveform, this will increase and decrease the height of the waveform. For the spectrogram, increasing the contrast will cause the calculated FFT values to be spread farther apart, thus increasing their range. When rendered, this will cause the output to use a wider range of values from the clut.

**Parameters:**

- **inValue**
  - An expression that evaluates to a number from 0.1 - 1000.
result A variable used to store the result in.

**Used in:** Spectrogram, Spectrum Analyzer, Waveform

**Display**
The following actions all apply to the extra visual items in the audio visualization tracks.

**QSSp_SetXScale(inScale) QSSp_GetXScale(result)**
Sets and gets the scale factor for time as displayed in the spectrogram and waveform. Valid ranges are from 1.0 to 16.0. Fractional values are ok. Scales that attempt to display more of the spectrogram than available will be pinned to an appropriate size. Numbers greater than 1.0 will have a zoom or enlarge effect.

**Parameters:**
inValue An expression that evaluates to a number from 1.0 to 16.0.
result A variable used to store the result in.

**Used in:** Spectrogram, Waveform

**QSSp_SetYScale(inScale) QSSp_GetYScale(result)**
Sets and gets the scale factor for the frequency as displayed in the spectrogram. Valid ranges are from 1.0 - 16.0. Fractional values are ok. At a scale factor of 1.0, the entire frequency range of the spectrogram will be displayed so that it exactly fits in the track. As the scale factor gets higher, less and less of the zoomed-in spectrogram will fit. You can then use the QSSP_SetYOffset to scroll the spectrogram vertically. If you then zoom out when scrolled, the scroll setting may get altered automatically such that no portion that is outside of the spectrogram will be displayed.

**Parameters:**
inValue An expression that evaluates to a number from 1.0 - 16.0.
result A variable used to store the result in.

**Used in:** Spectrogram

**QSSp_SetYAxisOffset(inValue) QSSp_GetYAxisOffset(result)**
Sets and gets the offset for the y axis in pixels. This has the effect of scrolling when scaled and allow you to zoom in on a particular portion of the spectrogram. Only useful when zoomed in using QSSP_SetYAxisScale. Valid values range from 0 to the track height.

**Parameters:**
inValue An expression that evaluates to a number from 0 to the track height.
result A variable used to store the result in.

**Used in:** Spectrogram

**Wired Actions**
QSSp_SetShowXAxis(inValue)
QSSp_GetShowXAxis(result)

Sets and gets the state of the x axis display. A state of 0, or false, means the axis is not shown.

**Parameters:**
- **inValue**: An expression that evaluates to true or false.
- **result**: A variable used to store the result in.

**Used in**: Spectrogram, Waveform

QSSp_SetShowYAxis(inValue)
QSSp_GetShowYAxis(result)

Sets and gets the state of the y axis display. A state of 0, or false, means the axis is not shown.

**Parameters:**
- **inValue**: An expression that evaluates to true or false.
- **result**: A variable used to store the result in.

**Used in**: Spectrogram, Waveform

QSSp_SetAxisBGColor(red,green,blue)
QSSp_GetAxisBGColor(outRed,outGreen,outBlue)

Sets and gets the background color for the axis display. Valid ranges for each color component are from 0 - 65535, where 0 means none or black.

**Parameters:**
- **red**: An expression that evaluates to a number from 0 - 65535.
- **green**: An expression that evaluates to a number from 0 - 65535.
- **blue**: An expression that evaluates to a number from 0 - 65535.
- **outRed**: A variable used to store the result in.
- **outGreen**: A variable used to store the result in.
- **outBlue**: A variable used to store the result in.

**Used in**: Spectrogram, Waveform

QSSp_SetAxisBorderColor(red,green,blue)
QSSp_GetAxisBorderColor(outRed,outGreen,outBlue)

Sets and gets the border color for the axis display. Valid ranges for each color component are from 0 - 65535, where 0 means none or black.

**Parameters:**
- **red**: An expression that evaluates to a number from 0 - 65535.
- **green**: An expression that evaluates to a number from 0 - 65535.
blue An expression that evaluates to a number from 0 - 65535.
outRed A variable used to store the result in.
outGreen A variable used to store the result in.
outBlue A variable used to store the result in.

Used in: Spectrogram, Waveform

QSSp_SetAxisTickColor(red,green,blue)
QSSp_GetAxisTickColor(outRed,outGreen,outBlue)

Sets and gets the axis ticks color for the axis display. Valid ranges for each color component are from 0 - 65535, where 0 means none or black.

Parameters:
red An expression that evaluates to a number from 0 - 65535.
green An expression that evaluates to a number from 0 - 65535.
blue An expression that evaluates to a number from 0 - 65535.
outRed A variable used to store the result in.
outGreen A variable used to store the result in.
outBlue A variable used to store the result in.

Used in: Spectrogram, Waveform

QSSp_SetAxisLabelColor(red,green,blue)
QSSp_GetAxisLabelColor(outRed,outGreen,outBlue)

Sets and gets the label color for the axis display. Valid ranges for each color component are from 0 - 65535, where 0 means none or black.

Parameters:
red An expression that evaluates to a number from 0 - 65535.
green An expression that evaluates to a number from 0 - 65535.
blue An expression that evaluates to a number from 0 - 65535.
outRed A variable used to store the result in.
outGreen A variable used to store the result in.
outBlue A variable used to store the result in.

Used in: Spectrogram, Waveform

QSSp_SetAxisLabelSize(inSize)
QSSp_GetAxisLabelSize(result)

Sets and gets the font size for the axis labels. Valid ranges are from 6 - 14.

Parameters:
inSize An expression that evaluates ton an integer from 6 - 14.
result A variable used to store the result in.

Used in: Spectrogram, Waveform

Wired Actions
QSSp_SetTimeCursorColor(red, green, blue)
QSSp_GetTimeCursorColor(outRed, outGreen, outBlue)

Sets and gets the color for the time cursor display. Valid ranges for each color component are from 0 - 65535, where 0 means none or black.

Parameters:
- **red**: An expression that evaluates to a number from 0 - 65535.
- **green**: An expression that evaluates to a number from 0 - 65535.
- **blue**: An expression that evaluates to a number from 0 - 65535.
- **outRed**: A variable used to store the result in.
- **outGreen**: A variable used to store the result in.
- **outBlue**: A variable used to store the result in.

Used in: Spectrogram, Waveform

QSSp_SetTimeCursorStyle(inStyle)
QSSp_GetTimeCursorStyle(result)

Sets and gets the style of the time cursor. Valid values are:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>QSSp_kTimeCursorNone</td>
<td>0</td>
</tr>
<tr>
<td>QSSp_kTimeCursorSolid</td>
<td>1</td>
</tr>
<tr>
<td>QSSp_kTimeCursorDotted</td>
<td>2</td>
</tr>
<tr>
<td>QSSp_kTimeCursorDashed</td>
<td>3</td>
</tr>
</tbody>
</table>

Parameters:
- **inValue**: An expression that evaluates to one of the cursor settings.
- **result**: A variable used to store the result in.

Used in: Spectrogram, Waveform

QSSp_SetHorizGrid(inValue)
QSSp_GetHorizGrid(result)

Sets and gets the display mode of the horizontal grid. The X axis must be visible in order to view the grid. Both the horizontal and vertical grids must be set to intersection to be able to see the intersection. Valid values are:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>QSSp_kGridOff</td>
<td>0</td>
</tr>
<tr>
<td>QSSp_kGridLines</td>
<td>1</td>
</tr>
<tr>
<td>QSSp_kGridDots</td>
<td>2</td>
</tr>
</tbody>
</table>
Constant       Value

QSSp_kGridIntersection       3

Parameters:
inValue       An expression that evaluates to one of the grid settings.
result       A variable used to store the result in.

Used in: Spectrogram, Waveform

QSSp_SetVertGrid(inValue)
QSSp_GetVertGrid(result)

Sets and gets the display mode of the vertical grid. The Y axis must be visible in order to view the grid. Both the horizontal and vertical grids must be set to intersection to be able to see the intersection. Valid values are:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>QSSp_kGridOff</td>
<td>0</td>
</tr>
<tr>
<td>QSSp_kGridLines</td>
<td>1</td>
</tr>
<tr>
<td>QSSp_kGridDots</td>
<td>2</td>
</tr>
<tr>
<td>QSSp_kGridIntersection</td>
<td>3</td>
</tr>
</tbody>
</table>

Parameters:
inValue       An expression that evaluates to one of the grid settings.
result       A variable used to store the result in.

Used in: Spectrogram, Waveform

QSSp_SetGridColor(red,green,blue)
QSSp_GetGridColor(outRed,outGreen,outBlue)

Sets and gets the grid color for the spectrogram and waveform display. Valid ranges for each color component are from 0 - 65535, where 0 means none or black.

Parameters:
red           An expression that evaluates to a number from 0 - 65535.
green         An expression that evaluates to a number from 0 - 65535.
blue          An expression that evaluates to a number from 0 - 65535.
outRed        A variable used to store the result in.
outGreen      A variable used to store the result in.
outBlue       A variable used to store the result in.

Used in: Spectrogram, Waveform
QSSp_SetCLUT(inTrackID, inTime)

Sets the clut for use in the spectrogram display. The clut will be taken from a 256 pixel wide track at the time specified in 600ths of a second. This track can be disabled so it does not draw in your main movie. It is easiest to make a clut using a color track and setting up a gradient that is rotated to 90 degrees. Passing in 0 for the track ID and 0 for the time will reset the clut to the default grey scale clut.

**Parameters:**
- inTrackID: An expression that evaluates to a valid track ID.
- inTime: An expression that evaluates to a valid time within that track.

**Used in:** Spectrogram

QSSp_SetBackgroundColor(red,green,blue)
QSSp_GetBackgroundColor(outRed, outGreen, outBlue)

Sets and gets the background color for the display. The spectrogram should rarely show this color as it will be seen only when there are not enough audio samples to fill the track display. Valid ranges for each color component are from 0 - 65535, where 0 means none or black.

**Parameters:**
- red: An expression that evaluates to a number from 0 - 65535.
- green: An expression that evaluates to a number from 0 - 65535.
- blue: An expression that evaluates to a number from 0 - 65535.
- outRed: A variable used to store the result in.
- outGreen: A variable used to store the result in.
- outBlue: A variable used to store the result in.

**Used in:** Spectrogram, Waveform, Spectrum Analyzer

QSSp_SetWaveFormColor(red,green,blue)
QSSp_GetWaveFormColor(outRed, outGreen, outBlue)

Sets and gets the waveform color for the display. Valid ranges for each color component are from 0 - 65535, where 0 means none or black.

**Parameters:**
- red: An expression that evaluates to a number from 0 - 65535.
- green: An expression that evaluates to a number from 0 - 65535.
- blue: An expression that evaluates to a number from 0 - 65535.
- outRed: A variable used to store the result in.
- outGreen: A variable used to store the result in.
- outBlue: A variable used to store the result in.

**Used in:** Waveform, Spectrum Analyzer

**Notes**
The following actions all apply to the notes as seen in the spectrogram and waveform visualizations.
QSSp_SetDisplayMode(inValue)
QSSp_GetDisplayMode(result)

Sets the display mode for the notes. Valid values:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>QSSp_kNoteDisplayNone</td>
<td>0</td>
</tr>
<tr>
<td>QSSp_kNoteDisplayNotesOnly</td>
<td>1</td>
</tr>
<tr>
<td>QSSp_kNoteDisplayNotesAndMetaData</td>
<td>2</td>
</tr>
</tbody>
</table>

**Parameters:**
inValue An expression that evaluates to one of the display constants.
result A variable used to store the result in.

**Used in:** Spectrogram, Waveform

QSSp_SetEditState(inValue)
QSSp_GetEditState(result)

Sets and gets the edit mode for the spectrograph. Valid values are:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>QSSp_kNoteEditNone</td>
<td>0</td>
</tr>
<tr>
<td>QSSp_kNoteEditCreation</td>
<td>1</td>
</tr>
</tbody>
</table>

When set to **QSSp_kNoteEditCreation** then the user will be able to click and drag out a selection in the waveform and spectrogram.

**Parameters:**
inValue An expression that evaluates to one of the edit state constants.
result A variable used to store the result in.

**Used in:** Spectrogram, Waveform

QSSp_SetGroupDisplayState(inGroupID, inValue)
QSSp_GetGroupDisplayState(inGroupID , variable)

The display state of the group will be set. Valid values are:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>QSSp_kGroupDisplayNone</td>
<td>0</td>
</tr>
<tr>
<td>QSSp_kGroupDisplayNormal</td>
<td>1</td>
</tr>
</tbody>
</table>
Parameters:
inValue An expression that evaluates to one of the display state constants.
result A variable used to store the result in.

Used in: Spectrogram, Waveform

QSSp_SetNoteDataFromXML(inXML)

All existing note objects and groups are deleted. The xml will be parsed and new note objects will be created. If there are errors in the xml, then parsing will stop at the first error detected. Here is some sample XML:

```xml
<Spectrum_Notes>
  <Notes>
    <One_Note>
      <Start_Time> 300 </Start_Time>
      <End_Time> 1000 </End_Time>
      <Min_Freq> 100 </Min_Freq>
      <Max_Freq> 8000 </Max_Freq>
      <Group_ID> 2 </Group_ID>
      <Note_Data>
        <Note_Data_ID>1000</Note_Data_ID>
        <Note_Data_String><![CDATA[This note is hidden]]></Note_Data_String>
        <Note_Data_Pos> 0 </Note_Data_Pos>
        <Note_Data_Size> 9 </Note_Data_Size>
        <Note_Data_Color>
          <red>0</red><green>0</green><blue>0</blue>
        </Note_Data_Color>
      </Note_Data>
    </One_Note>
    <One_Note>
      <Start_Time> 800 </Start_Time>
      <End_Time> 1900 </End_Time>
      <Min_Freq> 3000 </Min_Freq>
      <Max_Freq> 15000 </Max_Freq>
      <Group_ID> 1 </Group_ID>
      <Note_Data>
        <Note_Data_ID>1000</Note_Data_ID>
        <Note_Data_String><![CDATA[This is a sample]]></Note_Data_String>
        <Note_Data_Pos> 1 </Note_Data_Pos>
        <Note_Data_Size> 12 </Note_Data_Size>
      </Note_Data>
    </One_Note>
  </Notes>
</Spectrum_Notes>
```
Parameters:
inXML A string variable containing the XML to be parsed.

Used in: Spectrogram, Waveform

QSSp_AppendNoteDataFromXML(inXML)

The xml will be parsed and new note objects will be created and added to the existing notes database. Any existing groups or notes with the same IDs will be overwritten by data from the xml.
Parameters:
inXML A string variable containing the XML to be parsed.

Used in: Spectrogram, Waveform

QSSp_GetNoteDataAsXML(inGroupID, result)
Get all note objects or just those in a group as XML. If the group ID is 0, then all note objects will be returned. All group data is always returned.

Parameters:
inGroupID An integer expression evaluating to a valid group ID.
result A string variable that will be set to the xml.

Used in: Spectrogram, Waveform

QSSp_GetNoteAsXML(inNoteID, result)
Get all note data for the note passed in as XML. No group data for the note is returned.

Parameters:
inGroupID An integer expression evaluating to a valid note ID.
result A string variable that will be set to the xml.

Used in: Spectrogram, Waveform

QSSp_GetGroupsAsXML(result)
Get all group data for all notes as XML. No note data is returned.

Parameters:
result A string variable that will be set to the xml.

Used in: Spectrogram, Waveform

QSSp_SaveSelectionAsNote(result)
The variable will be filled in with the ID of a newly created note object, or 0. The user must have first dragged out a selection in the track. The track must also have its edit state set to QSSp_kNoteEditCreation.

Parameters:
result A variable that will be set to the note ID.

Used in: Spectrogram, Waveform

Wired Actions 58
QSSp_GetEventNoteID(result)

The variable will be filled in with the ID of the note object that triggered this event. Used only in event handlers for the spectrograph track.

**Parameters:**

- `result` A variable that will be set to the note ID.

**Used in:** Spectrogram, Waveform

QSSp_CountNotes(result)

The variable will be filled in with the current number of note objects.

**Parameters:**

- `result` A variable that will be set to the number of notes.

**Used in:** Spectrogram, Waveform

QSSp_ScaleToFitNote(inNoteID)

The scale factors, sample rate, step size etc will be adjusted so that the note fills the track.

**Parameters:**

- `inNoteID` An integer expression that evaluates to the note ID.

**Used in:** Spectrogram, Waveform

QSSp_SelectNote(inNoteID, inExtend)

The note object is selected. If `inExtend = false`, then all note objects are deselected first.

**Parameters:**

- `inNoteID` An integer expression that evaluates to the note ID.
- `inExtend` An integer expression that evaluates to true or false.

**Used in:** Spectrogram, Waveform

QSSp_DeselectNote(inNoteID)

The note object is deselected.

**Parameters:**

- `inNoteID` An integer expression that evaluates to the note ID.

**Used in:** Spectrogram, Waveform
QSSp_DeselectAllNotes
All note objects will be deselected.

*Used in:* Spectrogram, Waveform

QSSp_SelectNoteGroup(inGroupID, inExtend)
Selects all the notes in the group. If extend is false, then all notes will be deselected first.

**Parameters:**
- **inGroupID** An integer expression that evaluates to the group ID.
- **inExtend** An integer expression that evaluates to true or false.

*Used in:* Spectrogram, Waveform

QSSp_GetSelectedNote(result)
The variable will be filled in with the ID of the first selected note object.

**Parameters:**
- **result** A variable that will be set to the note ID.

*Used in:* Spectrogram, Waveform

QSSp_DeleteSelectedNotes
All selected note objects will be deleted.

*Used in:* Spectrogram, Waveform

QSSp_DeleteAllNotes
All note objects will be deleted.

*Used in:* Spectrogram, Waveform

QSSp_SetNoteGroup(inNoteID, inGroupID)  
QSSp_GetNoteGroup(inNoteID, result)
Sets the group that the note object belongs to. Note objects can only belong to ONE group. Group IDs are assigned automatically to new group objects when they are created.

**Parameters:**
inGroupID  An integer expression that specifies the new group ID for the note.
result    A variable that will be set to group ID of the note.

Used in: Spectrogram, Waveform

QSSp_SetNoteStartTime(inNoteID, inTime)
QSSp_GetNoteStartTime(inNoteID, result)
Sets the start time for the note object to the time in 600ths of a second. This time must be less than the end
time.

Parameters:
inNoteID  An integer expression that evaluates to the note ID.
inTime    An integer expression that evaluates to a valid time.
result    A variable that will be set to current time.

Used in: Spectrogram, Waveform

QSSp_SetNoteEndTime(inNoteID, inTime)
QSSp_GetNoteEndTime(inNoteID, result)
Sets the end time for the note object to the time in 600ths of a second. This time must be greater than the
start time.

Parameters:
inNoteID  An integer expression that evaluates to the note ID.
inTime    An integer expression that evaluates to a valid time.
result    A variable that will be set to current time.

Used in: Spectrogram, Waveform

QSSp_SetNoteMinFrequency(inNoteID, inFreq)
QSSp_GetNoteMinFrequency(inNoteID, result)
Sets the minimum frequency for the note object to the frequency in hz. This frequency must be greater than 0
and less than the maximum frequency allowed with the current FFT settings and less than the maximum set-
ting.

Parameters:
inNoteID  An integer expression that evaluates to the note ID.
inFreq    An integer expression that evaluates to a valid frequency.
result    A variable that will be set to current time.

Used in: Spectrogram, Waveform
QSSp_SetNoteMaxFrequency(inNoteID, inFreq)
QSSp_GetNoteMaxFrequency(inNoteID, result)

Sets the maximum frequency for the note object to the frequency in hz. This frequency must be greater than or equal to 0 and less than the maximum frequency allowed with the current FFT settings and greater than the minimum setting.

Parameters:
- **inNoteID** An integer expression that evaluates to the note ID.
- **inFreq** An integer expression that evaluates to a valid frequency.
- **result** A variable that will be set to current time.

**Used in:** Spectrogram, Waveform

QSSp_NudgeNoteStartTime(inNoteID, inNumPixels)

Alters the start time of the note object so that it visually moves by the number of pixels.

Parameters:
- **inNoteID** An integer expression that evaluates to the note ID.
- **inNumPixels** An integer expression that evaluates to the number of pixels to nudge by.

**Used in:** Spectrogram, Waveform

QSSp_NudgeNoteEndTime(inNoteID, inNumPixels)

Alters the end time of the note object so that it visually moves by the number of pixels.

Parameters:
- **inNoteID** An integer expression that evaluates to the note ID.
- **inNumPixels** An integer expression that evaluates to the number of pixels to nudge by.

**Used in:** Spectrogram, Waveform

QSSp_NudgeNoteMinFreq(inNoteID, inNumPixels)

Alters the minimum frequency of the note object so that it visually moves by the number of pixels.

Parameters:
- **inNoteID** An integer expression that evaluates to the note ID.
- **inNumPixels** An integer expression that evaluates to the number of pixels to nudge by.

**Used in:** Spectrogram, Waveform

QSSp_NudgeNoteMaxFreq(inNoteID, inNumPixels)

Alters the maximum frequency of the note object so that it visually moves by the number of pixels.

Parameters:
inNoteID An integer expression that evaluates to the note ID.
inNumPixels An integer expression that evaluates to the number of pixels to nudge by.

**Used in:** Spectrogram, Waveform

### QSSp_DeleteNote(inNoteID)

Deletes the note object.

**Parameters:**
inNoteID An integer expression that evaluates to the note ID.

**Used in:** Spectrogram, Waveform

### QSSp_GetNextNoteID(inNoteID, result)

QSSp_GetPreviousNoteID(inNoteID, result)

Returns the next note object’s ID, or the first one if the passed in ID = 0. The note IDs are returned in average time order. The average is the note start time and end times averaged.

**Parameters:**
inNoteID An integer expression that evaluates to the note ID.
result A variable that is set to the note ID.

**Used in:** Spectrogram, Waveform

### QSSp_GetNextVisibleNote(inNoteID, result)

QSSp_GetPreviousVisibleNoteID(inNoteID, result)

Returns the next visible note object’s ID, or the first one if the passed in ID = 0. The IDs are returned in numerical order.

**Parameters:**
inNoteID An integer expression that evaluates to the note ID.
result A variable that is set to the note ID.

**Used in:** Spectrogram, Waveform

### QSSp_ScrollToShowNote(inNoteID)

Time in the movie containing the audio track will be set such that the note ID will be centered in the visualization track.

**Parameters:**
inNoteID An integer expression that evaluates to the note ID.

**Used in:** Spectrogram, Waveform
QSSp_SetSelectedNoteColor(red, green, blue)
QSSp_GetSelectedNoteColor(outRed, outGreen, outBlue)

Sets and gets the color for notes that are selected.

**Parameters:**

- **red**
  - An expression that evaluates to a number from 0 - 65535.
- **green**
  - An expression that evaluates to a number from 0 - 65535.
- **blue**
  - An expression that evaluates to a number from 0 - 65535.
- **outRed**
  - A variable used to store the result in.
- **outGreen**
  - A variable used to store the result in.
- **outBlue**
  - A variable used to store the result in.

**Used in:** Spectrogram, Waveform

QSSp_SetNoteDefaultColor(red, green, blue)
QSSp_GetNoteDefaultColor(outRed, outGreen, outBlue)

Sets and gets the color for notes that do not have a color set.

**Parameters:**

- **red**
  - An expression that evaluates to a number from 0 - 65535.
- **green**
  - An expression that evaluates to a number from 0 - 65535.
- **blue**
  - An expression that evaluates to a number from 0 - 65535.
- **outRed**
  - A variable used to store the result in.
- **outGreen**
  - A variable used to store the result in.
- **outBlue**
  - A variable used to store the result in.

**Used in:** Spectrogram, Waveform

QSSp_MakeNewGroup(result)

The variable is set to the ID of a newly created group.

**Parameters:**

- **result**
  - A variable that is set to the group ID.

**Used in:** Spectrogram, Waveform

QSSp_DeleteGroup(inGroupID)

Deletes the group and all note objects in it.

**Parameters:**

- **inGroupID**
  - An integer expression that evaluates to the group ID.

**Used in:** Spectrogram, Waveform
QSSp_CountGroups(result)

The variable will be filled in with the current number of group objects.

Parameters:

- **result**
  - A variable that is set to the number of groups.

Used in: Spectrogram, Waveform

QSSp_GetSelectedGroup(result)

The variable will be filled in with the ID of the first selected note object's group.

Parameters:

- **result**
  - A variable that is set to the group ID.

Used in: Spectrogram, Waveform

QSSp_GetNextGroupID(inGroupID, result)
QSSp_GetPreviousGroupID(inGroupID, result)

The variable will be filled in with the ID of the next group object. To get the first group object, pass in 0 for the ID.

Parameters:

- **inGroupID**
  - An integer expression that evaluates to the group ID.

- **result**
  - A variable that is set to the next group ID.

Used in: Spectrogram, Waveform

QSSp_CountNotesInGroup(InGroupID , result)

The variable will be filled in with the number of notes in the group.

Parameters:

- **result**
  - A variable that is set to the number of notes in the group.

Used in: Spectrogram, Waveform

QSSp_GetNextNoteIDInGroup(inGroupID, inNoteID, result)
QSSp_GetPreviousNoteIDInGroup(inGroupID, inNoteID, result)

The variable will be filled in with the ID of the next note object in the group. To get the first note object, pass in 0 for the note ID. The note IDs are returned in average time order. The average is the note start time and end times averaged.

Parameters:
inGroupID     An integer expression that evaluates to the group ID.
inNoteID     An integer expression that evaluates to the note ID.
result       A variable that is set to the next note ID.

**Used in:** Spectrogram, Waveform

QSSp_SetNoteGroupColor(inGroupID, red, green, blue)
QSSp_GetNoteGroupColor(inGroupID, outRed, outGreen, outBlue)

Sets the color for the note display for all notes in that group. Valid ranges are from 0 - 65535.

**Parameters:**
- **red**     An expression that evaluates to a number from 0 - 65535.
- **green**   An expression that evaluates to a number from 0 - 65535.
- **blue**    An expression that evaluates to a number from 0 - 65535.
- **outRed**  A variable used to store the result in.
- **outGreen** A variable used to store the result in.
- **outBlue** A variable used to store the result in.

**Used in:** Spectrogram, Waveform

QSSp_SetNoteGroupData(inGroupID, inKey, inString)
QSSp_GetNoteGroupData(inGroupID, inKey, result)

Sets and gets the meta data identified by the key for the group to the string. The keys you use to identify meta data are up to you. We recommend that you use the same keys that are defined already for movie annotations.

**Parameters:**
- **inGroupID**     An integer expression that evaluates to the group ID.
- **inKey**          An integer expression that evaluates to a key that identifies the meta data.
- **result**         A string variable that is set to the meta data.

**Used in:** Spectrogram, Waveform

QSSp_SetNoteData(inNoteID, inKey, inString)
QSSp_GetNoteData(inNoteID, inKey, result)

Sets and gets the meta data identified by the key for the note object to the string. The keys you use to identify meta data are up to you. We recommend that you use the same keys that are defined already for movie annotations.

**Parameters:**
- **inNoteID**     An integer expression that evaluates to the note ID.
- **inKey**        An integer expression that evaluates to a key that identifies the meta data.
- **inString**     A string of information to be stored in the meta data.
- **result**       A string variable that is set to the meta data.

**Used in:** Spectrogram, Waveform
QSSp_SetNoteDataDisplayPos(inNoteID, inKey, inValue)
QSSp_GetNoteDataDisplayPos(inNoteID, inKey, result)

Sets the display position for the meta data identified by the key for the note object. If you want to show the note data only when the user mouses over the note, then add the constant QSSp_kDisplayPosMouseOver to the note position. Valid values are:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>QSSp_kDisplayPosHidden</td>
<td>0</td>
</tr>
<tr>
<td>QSSp_kDisplayPosTopLeftOutside</td>
<td>1</td>
</tr>
<tr>
<td>QSSp_kDisplayPosTopRightOutside</td>
<td>2</td>
</tr>
<tr>
<td>QSSp_kDisplayPosTopCenterOutside</td>
<td>3</td>
</tr>
<tr>
<td>QSSp_kDisplayPosTopLeftInside</td>
<td>4</td>
</tr>
<tr>
<td>QSSp_kDisplayPosTopRightInside</td>
<td>5</td>
</tr>
<tr>
<td>QSSp_kDisplayPosTopCenterInside</td>
<td>6</td>
</tr>
<tr>
<td>QSSp_kDisplayPosBotLeftOutside</td>
<td>7</td>
</tr>
<tr>
<td>QSSp_kDisplayPosBotRightOutside</td>
<td>8</td>
</tr>
<tr>
<td>QSSp_kDisplayPosBotCenterOutside</td>
<td>9</td>
</tr>
<tr>
<td>QSSp_kDisplayPosBotLeftInside</td>
<td>10</td>
</tr>
<tr>
<td>QSSp_kDisplayPosBotRightInside</td>
<td>11</td>
</tr>
<tr>
<td>QSSp_kDisplayPosBotCenterInside</td>
<td>12</td>
</tr>
<tr>
<td>QSSp_kDisplayPosMouseOver</td>
<td>64</td>
</tr>
</tbody>
</table>

**Parameters:**

- **inNoteID** An integer expression that evaluates to the note ID.
- **inKey** An integer expression that evaluates to a key that identifies the meta data.
- **inValue** An integer expression that evaluates to one of the position constants.
- **result** A variable that will be set to current position.

**Used in:** Spectrogram, Waveform

QSSp_SetNoteDataDisplaySize(inNoteID, inKey, inValue)
QSSp_GetNoteDataDisplaySize(inNoteID, inKey, result)

Sets and gets the font size for the meta data for the passed in note.

**Parameters:**

- **inNoteID** An integer expression that evaluates to the note ID.
- **inKey** An integer expression that evaluates to a key that identifies the meta data.
inValue            An integer expression that evaluates to a valid font size.
result             A variable that will be set to current font size.

Used in: Spectrogram, Waveform

QSSp_SetNoteDataDisplayColor(inNoteID, inKey, red, green, blue)
QSSp_GetNoteDataDisplayColor(inNoteID, inKey, outRed, outGreen, outBlue)

Sets and gets the color for the meta data for the passed in note.
Parameters:
inNoteID            An integer expression that evaluates to the note ID.
inKey               An integer expression that evaluates to a key that identifies the meta data.
red                  An expression that evaluates to a number from 0 - 65535.
green                An expression that evaluates to a number from 0 - 65535.
blue                 An expression that evaluates to a number from 0 - 65535.
outRed               A variable used to store the result in.
outGreen             A variable used to store the result in.
outBlue              A variable used to store the result in.

Used in: Spectrogram, Waveform

QSSp_SetNoteDataDefaultColor(red, green, blue)
QSSp_GetNoteDataDefaultColor(outRed, outGreen, outBlue)

Sets and gets the color for meta data that does not have a color set.
Parameters:
red                  An expression that evaluates to a number from 0 - 65535.
green                An expression that evaluates to a number from 0 - 65535.
blue                 An expression that evaluates to a number from 0 - 65535.
outRed               A variable used to store the result in.
outGreen             A variable used to store the result in.
outBlue              A variable used to store the result in.

Used in: Spectrogram, Waveform

Spectrum Analysis Settings

QSSp_SetFadeOut(inValue)
QSSp_GetFadeOut(result)

Sets and gets the fade out flag. If this flag is true, then the previous pixels drawn in the display are faded out rather than erased. With a fade opColor of about 10000, this can produce a pleasing effect.
Parameters:
**inValue**  An expression that evaluates to true or false.
**result**  A variable used to store the result in.

**Used in:** Spectrum Analysis

QSSp_SetFadeOpColor(inValue)
QSSp_GetFadeOpColor(result)

Sets and gets the fade opColor. The opColor controls the blend value used in fading. A value of 0 means there is no blending and a value of 65535 means that the previous pixels do not fade. Values in between specify various amounts of blending.

**Parameters:**
- **inValue**  An expression that evaluates to 0 - 65535.
- **result**  A variable used to store the result in.

**Used in:** Spectrum Analysis

QSSp_SetFadeColor(red, green, blue)
QSSp_GetFadeColor(outRed, outGreen, outBlue)

Sets and gets the fade color. When fade out is set to true, and there is a valid fade opColor set, then this will be the color that the display will fade into.

**Parameters:**
- **red**  An expression that evaluates to a number from 0 - 65535.
- **green**  An expression that evaluates to a number from 0 - 65535.
- **blue**  An expression that evaluates to a number from 0 - 65535.
- **outRed**  A variable used to store the result in.
- **outGreen**  A variable used to store the result in.
- **outBlue**  A variable used to store the result in.

**Used in:** Spectrum Analysis

QSSp_SetShowOnlyPeak(inValue)
QSSp_GetShowOnlyPeak(result)

Sets and gets the show only peak flag. If this flag is true, then the bar graph mode will show ONLY the peak bar and not the bars below that.

**Parameters:**
- **inValue**  An expression that evaluates to true or false.
- **result**  A variable used to store the result in.

**Used in:** Spectrum Analysis
QSSp_SetDecayRate(inValue)
QSSp_GetDecayRate(result)

Sets and gets the decay rate. When in bar graph mode this controls how fast the bars settle back down after
being pushed up due to a loud sound. A value of 10 produces a pleasing decay.

Parameters:
inValue  An expression that evaluates to 1 - 255.
result   A variable used to store the result in.

Used in: Spectrum Analysis

QSSp_SetRotation(inAngle, inX, inY)
QSSp_GetRotation(outAngle, outX, outY)

Sets and gets the rotation amount for the wave spectrum. Before each new wave is drawn, the previous wave
is rotated by this amount about the x,y position indicated. The angle can be specified in fractional amounts.

Parameters:
inAngle  An expression that evaluates to 0 - 360.
inX      An expression that evaluates to a pixel position.
inY      An expression that evaluates to a pixel position.
outAngle A variable used to store the angle in.
outX     A variable used to store the X position in.
outY     A variable used to store the Y position in.

Used in: Spectrum Analysis

QSSp_SetScale(scaleX, scaleY, inX, inY)
QSSp_GetScale(outScaleX, outScaleY, outX, outY)

Sets and gets the scale amount for the wave spectrum. Before each new wave is drawn, the previous wave is
scaled by this amount about the x,y position indicated. The scale can be specified in fractional amounts.

Parameters:
inScaleX An expression that evaluates to 0.001 - 10.0.
inScaleY An expression that evaluates to 0.001 - 10.0.
inX      An expression that evaluates to a pixel position.
inY      An expression that evaluates to a pixel position.
outScaleX A variable used to store the X scale in.
outScaleY A variable used to store the Y scale in.
outX     A variable used to store the X position in.
outY     A variable used to store the Y position in.

Used in: Spectrum Analysis
QSSp_SetTranslation(inX, inY)
QSSp_GetTranslation(outX, outY)

Sets and gets the translation amount for the wave spectrum. Before each new wave is drawn, the previous wave is translated by this amount.

**Parameters:**

- **inX**  
  An expression that evaluates to a pixel position.

- **inY**  
  An expression that evaluates to a pixel position.

- **outX**  
  A variable used to store the X position in.

- **outY**  
  A variable used to store the Y position in.

**Used in:** Spectrum Analysis

---

QSSp_SetZeroPoint(inValue)
QSSp_GetZeroPoint(result)

Sets and gets the zero position for the wave spectrum. When each new wave is drawn, this is the Y position of a wave with zero energy.

**Parameters:**

- **inValue**  
  An expression that evaluates to a pixel position.

- **result**  
  A variable used to store the result.

**Used in:** Spectrum Analysis

---

Other

---

QSSp_SetIdleRate(inValue)
QSSp_GetIdleRate(result)

Sets and gets the idle rate for the execution of idle actions. A value of -1 = off, any other number is the number of ticks to delay before the next execution of idle actions.

**Parameters:**

- **inValue**  
  An expression that evaluates to a number from -1 on up.

- **result**  
  A variable used to store the result in.

**Used in:** Spectrogram, Waveform, Spectrum Analyzer

---

QSSp_GetRenderingProgress(result)

When rendering the full spectrogram or waveform, this will return the rendering progress as a percentage. This value is only valid when scale to fit is set to fit all, otherwise it indicates the progress of just a portion of the rendering.

**Parameters:**
result A variable used to store the result in.

Used in: Spectrogram, Waveform

QSSp_GetMousePosTime(inValue, result)
Returns the time represented at the horizontal pixel position passed in.

Parameters:
inValue An expression indicating a pixel position in the track.
result A variable used to store the result in.

Used in: Spectrogram, Waveform

QSSp_GetMousePosFrequency(inValue, result)
Returns the frequency represented at the vertical pixel position passed in.

Parameters:
inValue An expression indicating a pixel position in the track.
result A variable used to store the result in.

Used in: Spectrogram