WHEATSTREAM & STREAMBLADE

Setup and Operation Manual

Version 2.0
August 2023

Wheatstone Corporation
600 Industrial Drive
New Bern, NC 28562
USA

Tel (252) 638-7000

Email techsupport@wheatstone.com

Web www.wheatstone.com

Attention!

Federal Communications Commission (FCC) Compliance Notice:
Radio Frequency Notice

NOTE: This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

⚠ **This is a Class A product. In a domestic environment, this product may cause radio interference, in which case, the user may be required to take appropriate measures.**

This equipment must be installed and wired properly in order to assure compliance with FCC regulations.

Caution! ***Any modifications not expressly approved in writing by Wheatstone could void the user's authority to operate this equipment.***

CE

RoHS Compliant 2002/95/EC

CONTENTS

A NOTE FROM THE DESIGNER OF STREAMBLADE'S AUDIO PROCESSING

Back in the late sixties during my first "radio engineering job" I became fascinated by audio processing. It soon became a hobby (some claim a curse) and over the ensuing years I built many custom (and secret) audio processors for stations where I worked. Over those years I also became familiar with just about every other audio processor made, so when I state that Streamblade's audio processing is like nothing that's existed before, I don't do it lightly.

When I was asked to design the audio processing portion of Wheatstone's new Streamblade product my primary goal was to create something very powerful, but also something you won't notice while it's doing all the heavy lifting it needs to do to keep codecs happy. Not a trivial challenge…

Fortunately, I'd been experimenting at length with Neural Networks and discovered that it was indeed possible for the type of processing to be modified on the fly to be exactly what is needed at that instant – no more and no less. Therefore, gone are the usually fixed Attack, Release, Ratio and Threshold controls; each band's Neural Network handles those parameters on the fly by learning the characteristics of the current program type *at that instant*, and then adapts them so each band always does exactly what is needed, *at that instant*.

Also gone are the always troublesome to adjust inter-band coupling controls; the Neural Networks "know" what the desired spectral balance should be based on how the user has set the Band Mix controls and it maintains that balance as the program content changes.

One of each band's Neural Network tasks is to maintain historical data about the current program type, data it then uses to further hide the action of controlling levels. The end result is that the "sound" of Streamblade's processing, dynamics-wise, sounds very close to that of the original unprocessed audio. Even though the dynamic range of the incoming program has been greatly reduced, there remains a sense of dynamics whose sole purpose is to enhance long-term stream listening by minimizing myriad factors that contribute to listener fatigue.

The audio can be loud but is never "squashed", though a special Density control *does* enable the user to increase the audibility of processing if that is the goal. Likewise a special Speed control governs the longish-term dynamics behavior by modifying how the Neural Networks use their historical data. The overall processing can be *quite* aggressive at controlling levels, but without ever sounding busy, smashed or lifeless.

Several dozen Factory Presets allow the end user to experiment with Streamblade's various ways of creating dynamic and spectral signatures. The user is strongly encouraged to take some time to listen to *all* of the Factory presets to get a sense of what Streamblade's processing is really capable of. Enjoy!

Jeff Keith

Jeff Keith, CPBE, NCE
Senior Product Development Engineer
Wheatstone Corporation

CHAPTER 1 – DEVICE SETUP

INTRODUCTION

Congratulations on your purchase of this Wheatstone streaming product! You're minutes away from delivering the best sounding streams to your audience's ears, wherever they chose to listen to you.

We recognize that you're anxious to get this online right away, and after reading through this section, you should be able to get your Wheatstream online, connected to your stream host, and after choosing a preset having your audio processed in a way that best suits your content.

However, we hope you'll stick around for the whole manual to learn how to best choose your codec and bitrate and to learn the important differences between processing audio online versus on air. This chapter will cover the nuts and bolts of getting online, and the chapter following will teach you how to optimize your stream to keep your audience listening longer.

WHEATSTREAM VS STREAMBLADE

Wheatstone Corporation offers two streaming devices which are essentially similar.

The Wheatstream is an 8 channel streaming appliance which operates exclusively on AoIP (Audio over IP) stream.

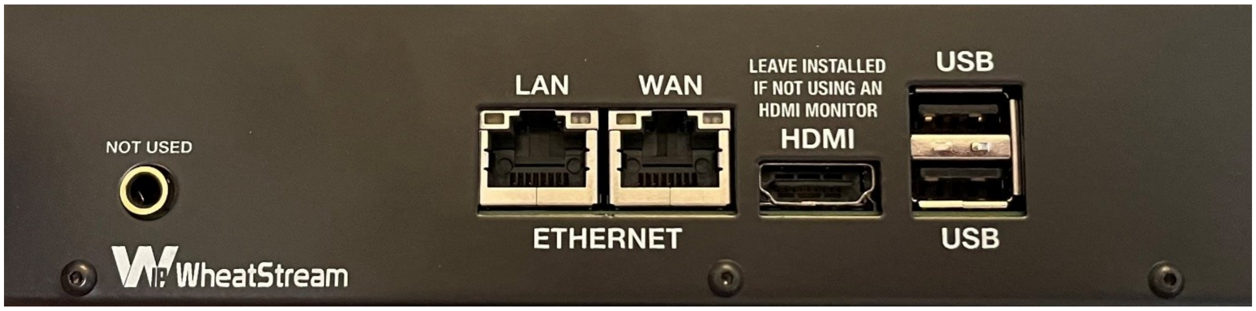
The Streamblade is the same 8 channel streaming appliance as the Wheatstream, but adds an 8 input/8 output WheatNet-IP blade with analog and digital I/O for stations that do not have a full AoIP infrastructure or stations that need additional I/O for their WheatNet-IP system.

This manual applies to both products and covers the setup and operation of the streaming system. If you have a Streamblade, it is important to understand that the A/D Blade and the enclosed streaming encoder are two separate entities, which, despite sharing the same box, do not directly interact with or even know about each other. They will not even share the same IP address. Thus you will configure the Blade side, with its analog/digital I/O, as you would any other Blade in your WheatNet-IP system, using WheatNet-IP Navigator, and you will configure the streaming side as described in the present manual. Blade setup and configuration is covered in a separate document.

Henceforth in this manual the name Wheatstream will be used to refer to both Streamblade and Wheatstream devices. Comments or instructions specific to one or the other are indicated as such

INITIAL SETUP

The Wheatstream device runs on an embedded computer. Make the following connections before applying power:



LAN: This is the connection to the WheatNet-IP network or AES-67 audio network.

WAN: This is the connection to the internet.

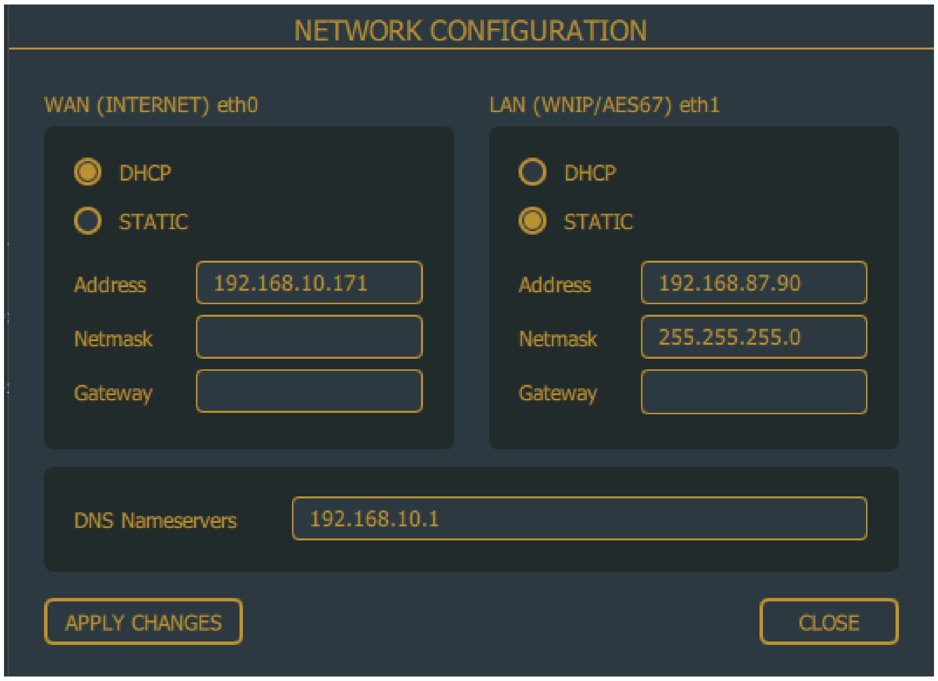
HDMI: Connect a HDMI monitor here for initial configuration. After setting up the network, you can optionally use the remote GUI on a separate computer to administer the Wheatstream and disconnect this monitor. If you choose to do this, insert the dummy HDMI plug into the port, as the embedded PC will not boot if this jack is empty.

USB: Connect a keyboard and mouse here for the initial configuration.

Now you can apply power to the device. The embedded PC will beep once upon booting. After the initial boot process you will be taken to the main screen.

CONFIGURE THE NETWORKS

We need to configure the IP address of both the AoIP network and connection to the internet. In the lower left corner of the screen, click the Network button to open the Network Configuration dialog.



NETWORK CONFIGURATION

WAN (INTERNET) eth0

[✓ DHCP] STATIC

Address: 192.168.10.171

Netmask:

Gateway:

LAN (WNIP/AES67) eth1

DHCP [✓ STATIC]

Address: 192.168.87.90

Netmask: 255.255.255.0

Gateway:

DNS Nameservers: 192.168.10.1

On the WAN side, choose DHCP or Static to match what your network requires. If you choose DHCP, the DNS nameserver(s) and the IP address assigned via DHCP will automatically fill in after you apply the changes. If you are using a static IP, you will need to fill in the address, netmask, gateway, and DNS server information.

On the LAN side, you will almost invariably choose a Static address. Most WheatNet-IP networks use static IPs, so fill in the address and netmask, taking care not to use an address in use by another device on the network. Typically there is no gateway on the LAN side.

Click Apply Changes, then Close.

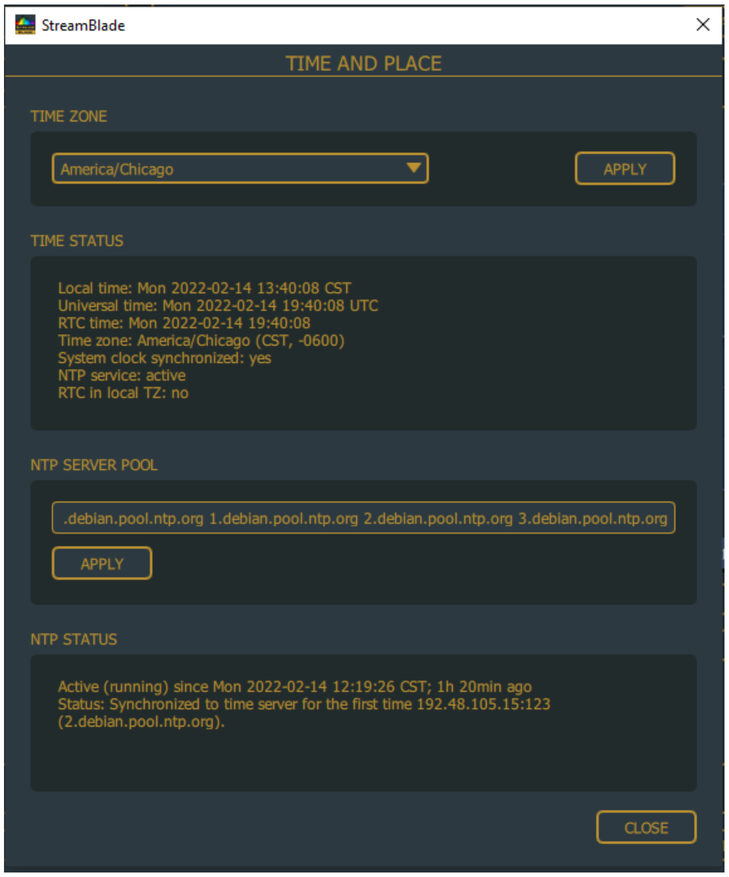
In rare instances a custom configuration is required which cannot be created using the Network Configuration dialog. Chapter 8 contains guidance on advanced network setup and DHCP configuration.

Streamblade Specific

Streamblade units have three network connections; two of these connections are on the WNIP network and one is to the internet. One WNIP connection is for getting audio in and out of the blade hardware and the other WNIP connection is for joining the embedded streaming PC to the WNIP network. The important thing to know is that you want to assign two unique static IP addresses on your AoIP LAN to those WNIP connections (assigning the same IP to both will NOT work) and then you will set up the WAN connection to see the internet.

SETTING THE TIME ZONE

Click the TIMEZONE button in the lower left corner to set up the proper time zone and connect to NTP



America/Chicago

```
Local time: Mon 2022-02-14 13:40:08 CST
Universal time: Mon 2022-02-14 19:40:08 UTC
RTC time: Mon 2022-02-14 19:40:08
Time zone: America/Chicago (CST, -0600)
System clock synchronized: yes
NTP service: active
RTC in local TZ: no
```

.debian.pool.ntp.org 1.debian.pool.ntp.org 2.debian.pool.ntp.org 3.debian.pool.ntp.org

```
Active (running) since Mon 2022-02-14 12:19:26 CST; 1h 20min ago
Status: Synchronized to time server for the first time 192.48.105.15:123
(2.debian.pool.ntp.org).
```

Choose the proper city for your region and click Apply. (Tip: if you are located in a place that does not observe Daylight Saving Time, choose a city that reflects that. For example, someone in Arizona should pick Phoenix and not Denver, because while both cities are in the Mountain Time Zone, Denver observes DST and Phoenix does not.)

In the NTP Server Pool field, several publicly accessible NTP servers are already listed as default. If your network requires you to connect to a private NTP server, enter it here and click Apply.

Setting the correct time zone and setting up NTP will ensure that your logs will have accurate times and scheduled events will happen at the correct time.

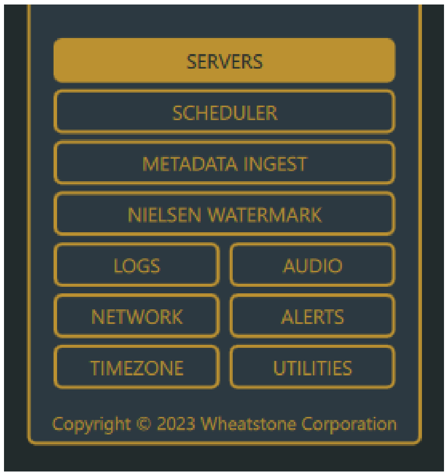
CONFIGURING THE REMOTE SOFTWARE

Let's face it, this device is most likely to end up in a rack room, and rack rooms are loud and cold and generally not a comfortable place to set up processing.

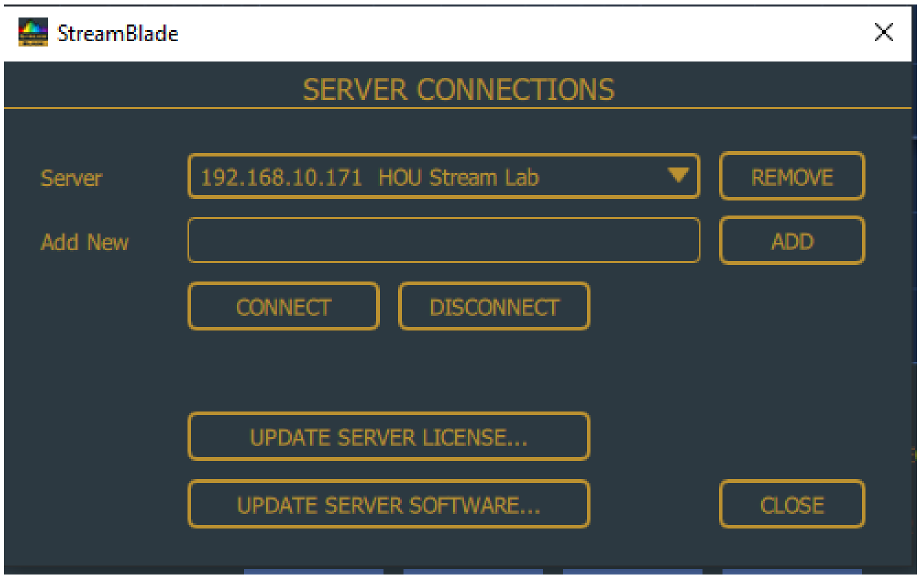
That's why after you've done the initial setup of the Wheatstream device, you can remote into the box from anywhere on your network. Why stand in the rack room with a keyboard balancing on a spool of wire when you can manage Wheatstream from your desk?

Install Streamblade Remote on a PC that is on the same network/subnet as one of the Wheatstream's network connections. It can be either the WAN (Internet) or LAN (AoIP) so long as your remote PC can connect to the device.

After installation, start the program, and click on the Servers button on the left side.



If your unit doesn't already appear in the Server dropdown list, type the IP address for the WAN or LAN port of your Wheatstream device in the Add New field, then click Add. You will automatically connect.



If you manage multiple Wheatstream appliances, as you add new devices they will be saved in the drop-down menu until you remove them. Using this feature you can then switch to other servers simply by selecting them from the menu.

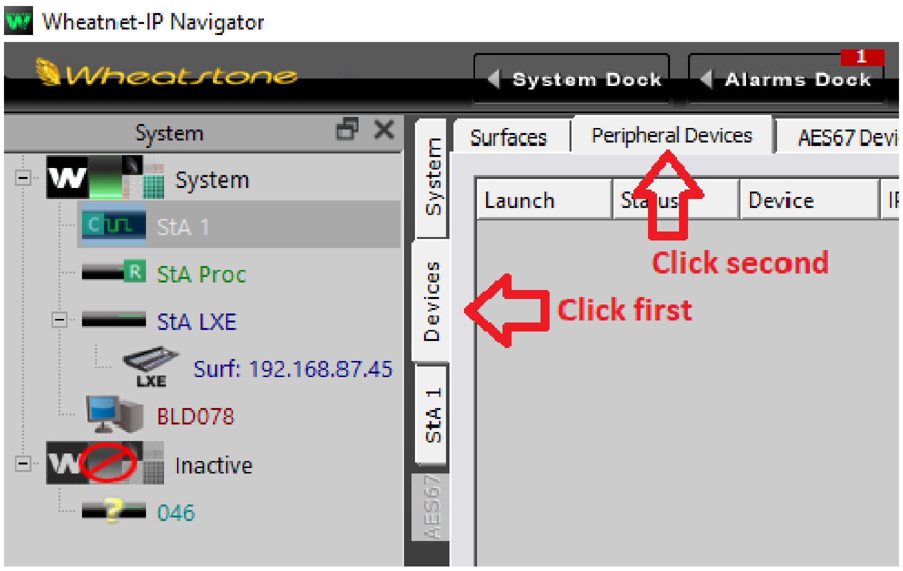
JOINING THE WHEATSTREAM TO THE WHEATNET-IP NETWORK

Wheatstream is configured as a peripheral device to the WheatNet-IP network. Before we proceed any further, we need to connect the streaming system to a host blade using WheatNet-IP Navigator.

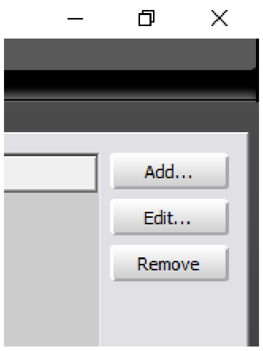
If you do not already have a copy of Navigator, contact Wheatstone support. Navigator is necessary to route audio in and out of the device and will connect to the same network as the LAN connection on the Wheatstream. You will find it most convenient to have two network cards on your administration PC; one that's connected to the internet network and one that's connected to the audio network.

For Wheatstream, you can assign any Blade in the system as the host. For Streamblade specifically, you will usually choose the integral Blade as the host, but you could assign any Blade in the system as its host.

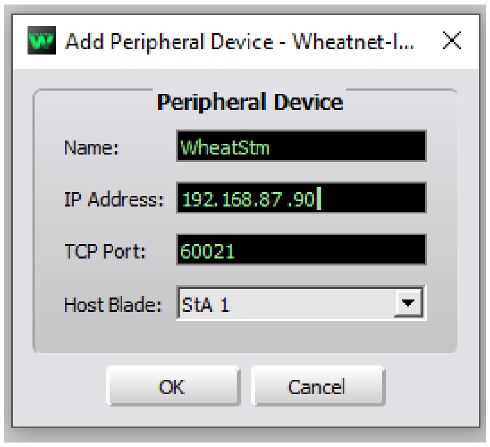
After you have added the streamer as a peripheral device, the sources and destinations for the streamer will appear with the host Blade, so choose a host Blade that's easy for you to find and logically makes sense to you. Got your blade in mind? Great, let's start by opening Navigator and click on the Devices tab on the left, then the Peripheral Devices tab up top.



Now, on the far right hand side of the screen, click Add.



In the center of the screen you'll get a box to fill in:



Add Peripheral Device - Wheatnet-I...

Peripheral Device

Name:	WheatStm
IP Address:	192.168.87.90
TCP Port:	60021
Host Blade:	StA 1

OK Cancel

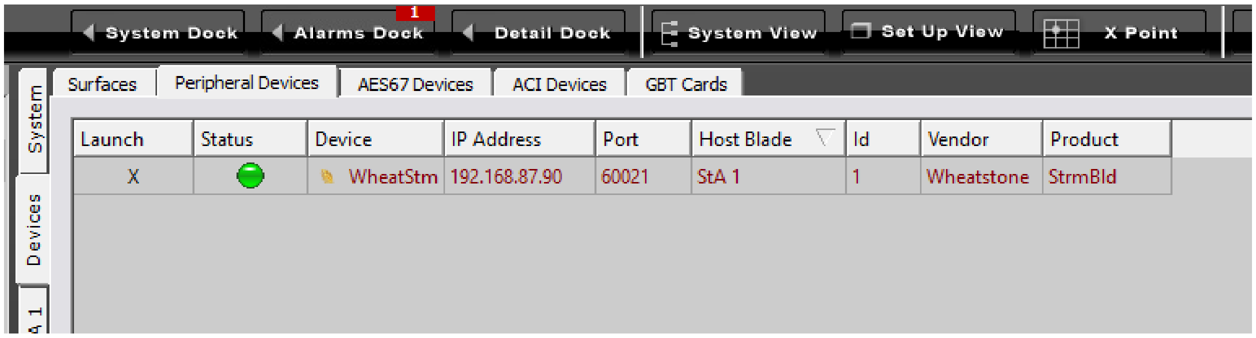
For the Name, give your streamer an 8-character name that will help you identify what it is in Navigator.

For the IP Address, enter the address from the WNIP side of the network settings you just set.

Leave TCP Port 60021 unchanged; this is the port Navigator uses to communicate with the box.

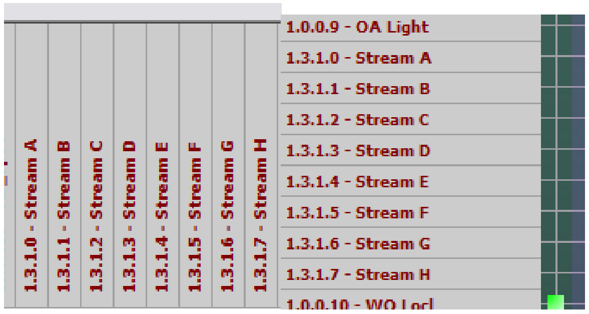
Use the Host Blade drop-down to assign the Wheatstream/Streamblade streaming device to the desired Blade.

Click OK. As long as the Wheatstream is powered up, connected to the correct network switches, and has the correct network settings, you will see a green light.



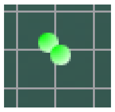
Launch	Status	Device	IP Address	Port	Host Blade	Id	Vendor	Product
X	●	WheatStm	192.168.87.90	60021	StA 1	1	Wheatstone	StrmBld

Now, you're good to go, and when you click on the System/Crosspoints tabs in Navigator, you will see Stream A – Stream H in the sources and destinations.



ROUTING AUDIO TO THE STREAMS

From the System/Crosspoints tab, click where the crosspoints intersect between the source along the top (a Program output from a studio or the output of an automation PC, whatever you are feeding to the stream) to the destination of Stream A – Stream H.



Tip! If nothing happens when you click in the intersection of the lines, then Navigator may be in safety mode where you must hold down the Ctrl key on the keyboard while clicking.

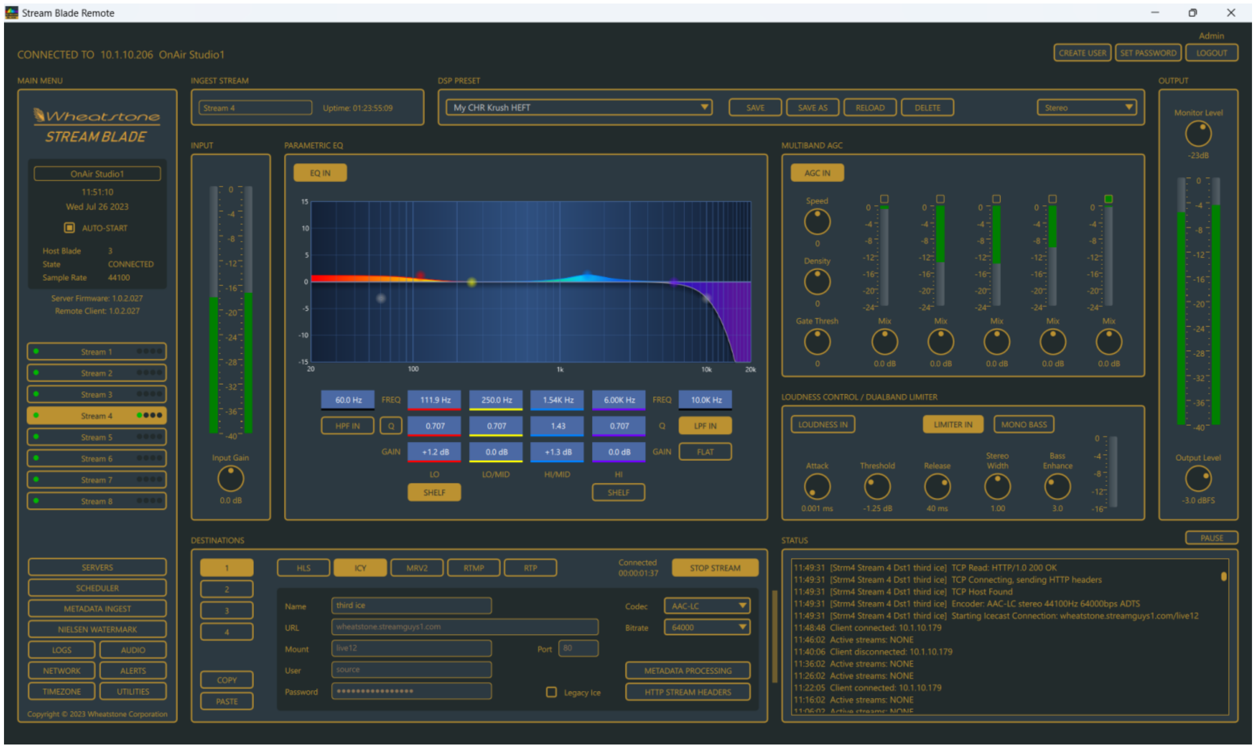
If audio is present, Navigator will show a green dot (mono) or green pair of dots (stereo), occasionally blinking red at peaks in the audio. If audio is not present (or very low), the dot(s) will be blue.

Back at the Wheatstream GUI, you will see the audio on the meters.

Likewise, back in Navigator, you can find the output of the stream you are working on (Stream A – Stream H) and route it to an output that feeds some monitor speakers so you can evaluate your work.

TOUR OF THE MAIN WINDOW

Before we go any further, let's take a quick tour of the main GUI so that you are oriented better for the discussions ahead.



CONNECTED TO 10.1.10.206 OnAir Studio1

OnAir Studio1
11:51:10
Wed Jul 26 2023
AUTO-START
Host Blade 3
State CONNECTED
Sample Rate 44100
Server Firmware: 1.0.2.027
Remote Client: 1.0.2.027
Copyright © 2023 Wheatstone Corporation

Stream 4 Uptime: 01:23:55:09

My CHR Krush HEFT

Connected 00:00:01:37

Name: third ice
URL: wheatstone.streamguys1.com
Mount: live12
Port: 80
User: source
Password: ••••••••••••••••
Codec: AAC-LC
Bitrate: 64000

```
11:49:31  [Strm4 Stream 4 Dst1 third ice]  TCP Read: HTTP/1.0 200 OK
11:49:31  [Strm4 Stream 4 Dst1 third ice]  TCP Connecting, sending HTTP headers
11:49:31  [Strm4 Stream 4 Dst1 third ice]  TCP Host Found
11:49:31  [Strm4 Stream 4 Dst1 third ice]  Encoder: AAC-LC stereo 44100Hz 64000bps ADTS
11:49:31  [Strm4 Stream 4 Dst1 third ice]  Starting Icecast Connection: wheatstone.streamguys1.com/live12
11:48:48  Client connected: 10.1.10.179
11:46:02  Active streams: NONE
11:40:06  Client disconnected: 10.1.10.179
11:36:02  Active streams: NONE
11:26:02  Active streams: NONE
11:22:05  Client connected: 10.1.10.179
11:16:02  Active streams: NONE
11:06:02  Active streams: NONE
```

The GUI is divided into a number of panes, each one labeled according to its function. Starting at the top of the Main Menu on the left side, we have device name, time and date, the Auto-Start option (enabled by default, this function ensures that your inbound and outbound streams will be started automatically after a reboot), WNIP host information, current software version, ingest stream selection, and finally at the bottom you have access various auxiliary functions.

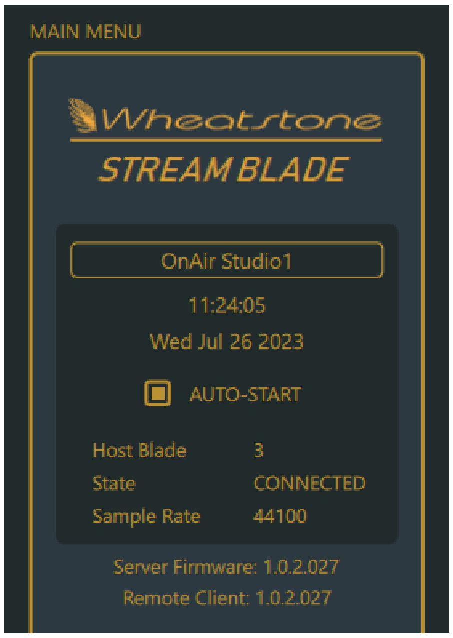
The top portion of the remaining GUI is primarily devoted to controls for processing of the ingest stream: input and output meters, EQ, AGC, and Limiter, DSP preset selection. Below the EQ controls we have the Destinations pane (where your outbound streams are configured), and in the lower right corner the Status pane with its continual updates.

The Status pane displays ongoing status messages, error messages, incoming and outgoing metadata, etc. Note the "grab bar" between the Destination and Status panes which lets you drag the Status pane to the left (making it wider), which is useful for reading lengthier messages. The Pause button at the top right corner of the Status pane stops display of new messages (allowing to you read existing messages without them disappearing off the bottom of the display every time a new message comes in). Watch this display long enough and you'll notice a status summary is posted every 10 minutes listing active ingest and outbound streams.

Everything in the Status display is written to the Logs, so if you miss something here you'll be able to find it later in the logs. The Status display is, in fact, simply the most recent 300 lines of the current log.

NAMING WHEATSTREAM DEVICES

You'll notice in the Wheatstream GUI there's a place above the clock where you can name your device.



OnAir Studio1
11:24:05
Wed Jul 26 2023

Host Blade	3
State	CONNECTED
Sample Rate	44100

Server Firmware: 1.0.2.027
Remote Client: 1.0.2.027

Click in the box to edit the text. Press Enter when finished.

When you are managing more than one machine from the Remote GUI, it's nice to have a friendly name to identify which unit you're looking at.

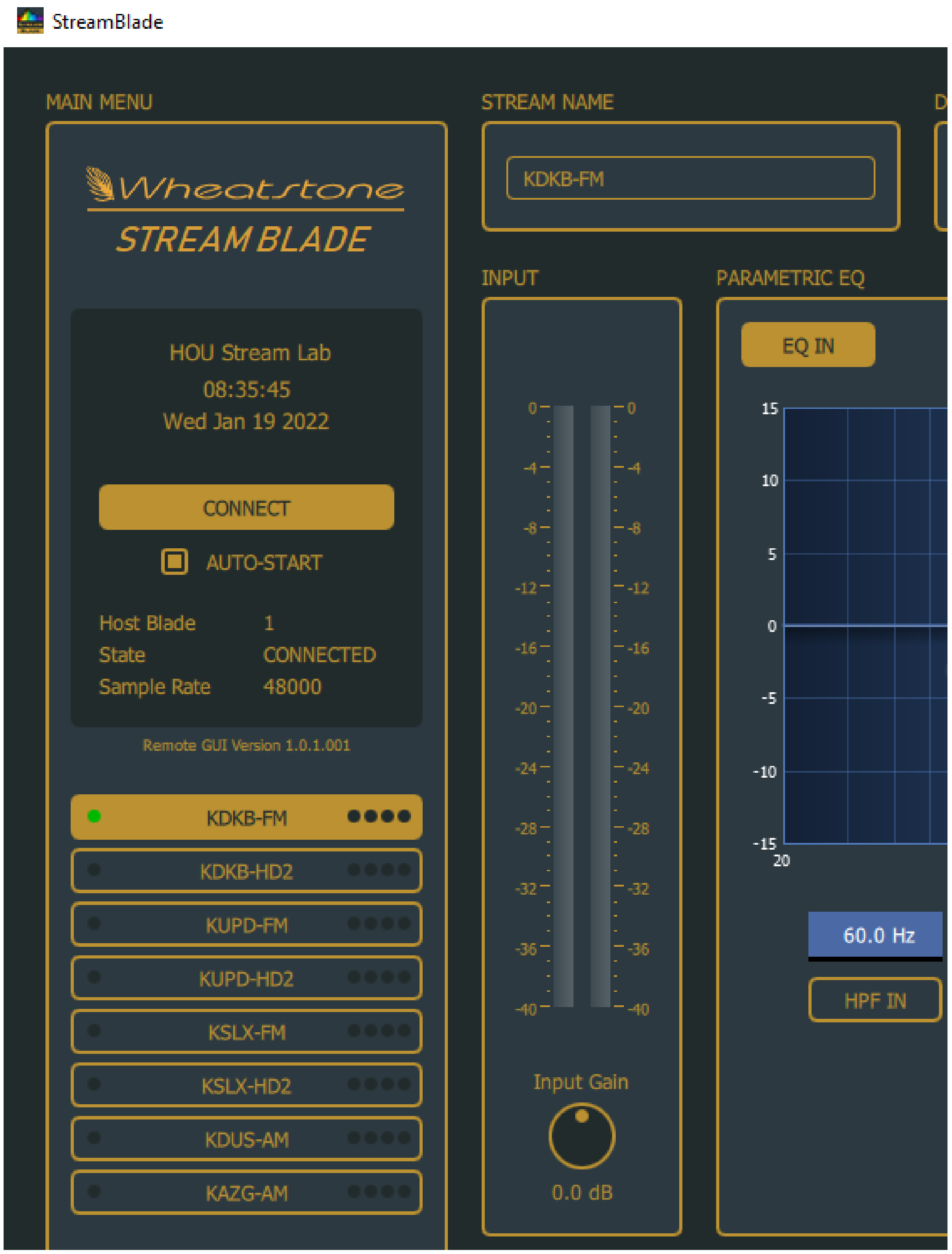
STREAMBLADE SPECIFIC

Note that the naming field here is not the same as the name of the companion Blade that you set in Navigator. You can give them the same name, or different ones depending upon your needs.

NAMING THE STREAMS

Likewise, it's nice to look at the call letters or program service instead of Stream 1 – Stream 8.

In the left menu, click the button for the stream you wish to name. In the Stream Name box up top, type the new name. When you press Enter the change will take effect.



Stream Name: KDKB-FM

HOU Stream Lab
08:35:45
Wed Jan 19 2022

Host Blade	1
State	CONNECTED
Sample Rate	48000

KDKB-FM

60.0 Hz

0.0 dB

GAIN ADJUSTMENTS

Wheatstream gives you the ability to adjust the input, output, and monitor gain of each stream.

INPUT GAIN

Adjust the input gain knob to set an appropriate level. The metering is peak, not average. Levels should hit between -12 and 0 without exceeding 0.

OUTPUT GAIN

Adjust the Output Level to set the gain feeding the stream encoder. Changes made to this knob will be reflected in the metering. As with the input, this meter displays peak, not average and levels should hit between -12 and 0 without exceeding 0. The default level set to -3

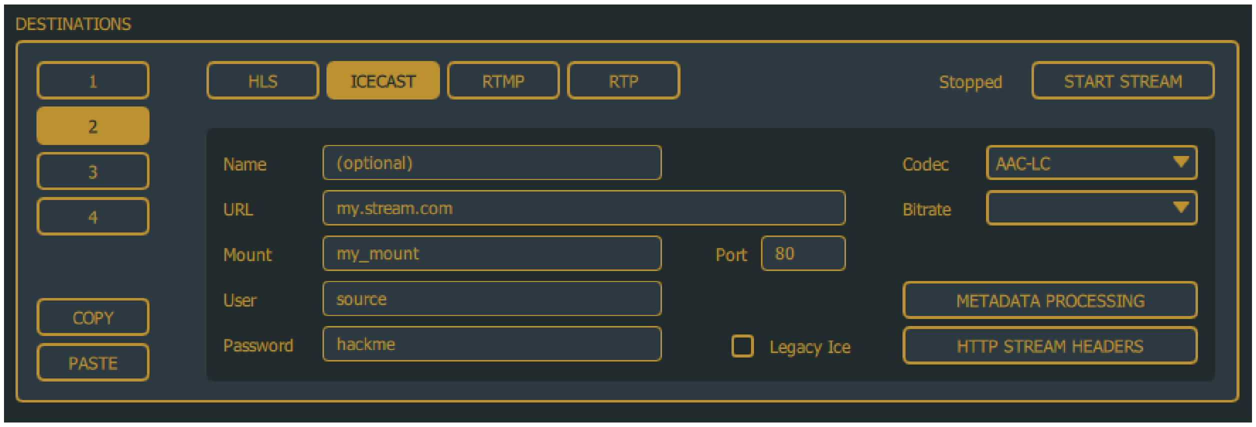
MONITOR LEVEL

Located just above the output meter, this control will adjust the levels of the monitor outputs (those listed as Stream A-Stream H in Navigator). Changes to this control will not be reflected in the output meter, but you will hear them in your monitors. This control allows you to connect a powered speaker to a Blade output and have a level control on the screen to adjust the volume.

STREAM PROVIDER SETUP

In Wheatstream, each ingest stream can be sent to as many as four different destinations, each with its own DSP processing, codec, bitrate, and transport format. The destination is typically a CDN (content distribution network), though RTP streams (either linear PCM or compressed AAC) may be sent to any receiver, such as a transmitter, which accepts that format. Your CDN will supply you with connection information such as the URL, user credentials, preferred transport format and bitrate, etc. Be aware that once a destination stream is started, none of its attributes (codec, bitrate, transport) may be changed without stopping the stream, setting the new parameters, then restarting the stream.

Select the current ingest stream on the left hand side of the screen, then in the Destinations section, click on the output number you wish to edit.



Name (optional)
URL my.stream.com
Mount my_mount
Port 80
User source
Password hackme
Legacy Ice
Codec AAC-LC
Bitrate

Choose the transport type that your provider supports: HLS, Icecast, RTMP or MRV2 (Triton Digital), and enter your credentials. Note that you may give each of your outbound streams a name for reference purposes. Specific instructions for connecting to servers at Triton Digital are found in Chapter 6.

Legacy Icecast
When Icecast was first introduced in 1999, it employed an HTTP method ("SOURCE") which has since been deprecated. However, many of these older servers are still in use, and they still expect the older method to be used in the establishment of the initial connection to the stream server.

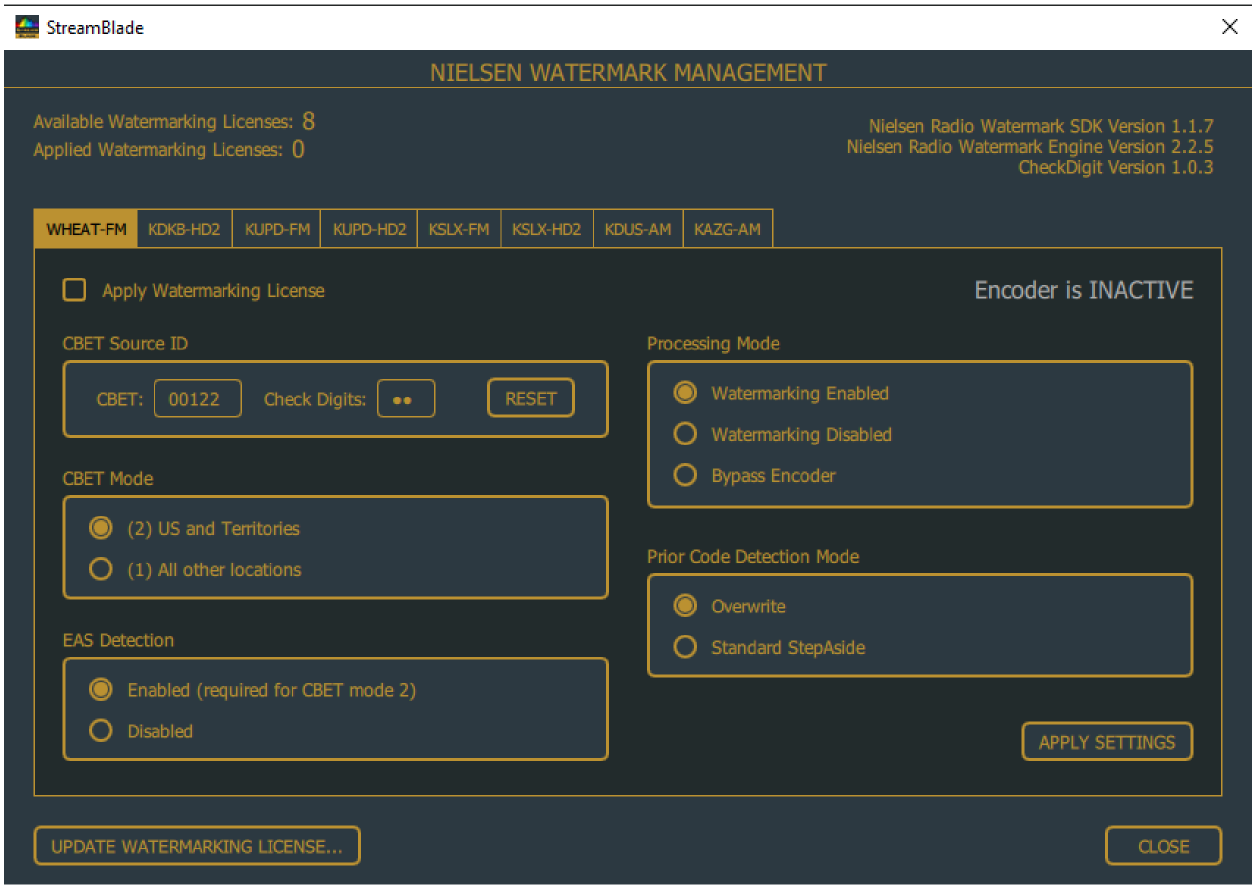
Metadata Processing
A very important part of Wheatstream's job is to forward metadata (typically from your automation system) to the CDN, where it is used to display artist, title and other information to the listener. Metadata also includes signals that are used to trigger ad replacements for streams which are being heard outside of the geographical location of the stream's origin. It is important to understand that there are no universally accepted standards for the transmission or formatting of metadata – neither for the sources which generate it, nor for the CDN servers which ingest it. In other words, metadata may arrive in virtually any format, and then it needs to be massaged into a different format acceptable to the CDN. For this reason, Wheatstream employs transformation filters written in the Lua scripting language. These filters are capable of ingesting metadata in any format, then spitting it back out in another format for the CDN.

Wheatstream can accept metadata coming in via UDP or TCP. Click the METADATA INGEST button to open a dialog where TCP/UDP reception can be enabled. In the DESTINATIONS pane, click the METADATA PROCESSING button to switch the DSP display to the Metadata Processing display. A small set of transformation filters are provided, any of which may be modified and customized to your (and the CDN's) needs. Typically there is a process of discovery and negotiation with the CDN over the precise format of the metadata update messages that will be sent to it. Adventurous users will be able to write or modify transform filters themselves, but we at Wheatstone are also ready and willing to help with this task. Metadata and ad-insertion signaling are areas where commercial standards are actively evolving, so you can expect to see many changes in this area in years to come.

For Icecast streams, please note that static metadata such as the name of the station, description and genre can be set by clicking the HTTP STREAM HEADERS button. Again, these static headers must be set before the destination stream is started.

A detailed primer on metadata is provided in Chapter 3.

NIELSEN WATERMARKING
The Wheatstream processor contains a Nielsen PPM encoder to ensure that your station gets ratings credit for panelists carrying meters that hear your streams. Click the Nielsen Watermark button.


StreamBlade

NIELSEN WATERMARK MANAGEMENT

Available Watermarking Licenses: 8
Applied Watermarking Licenses: 0

Nielsen Radio Watermark SDK Version 1.1.7
Nielsen Radio Watermark Engine Version 2.2.5
CheckDigit Version 1.0.3

☐ Apply Watermarking License

Encoder is INACTIVE

CBET Source ID

CBET: 00122 Check Digits: •• RESET

Processing Mode

[✓ Watermarking Enabled] Watermarking Disabled Bypass Encoder

CBET Mode

[✓ (2) US and Territories] (1) All other locations

Prior Code Detection Mode

[✓ Overwrite] Standard StepAside

EAS Detection

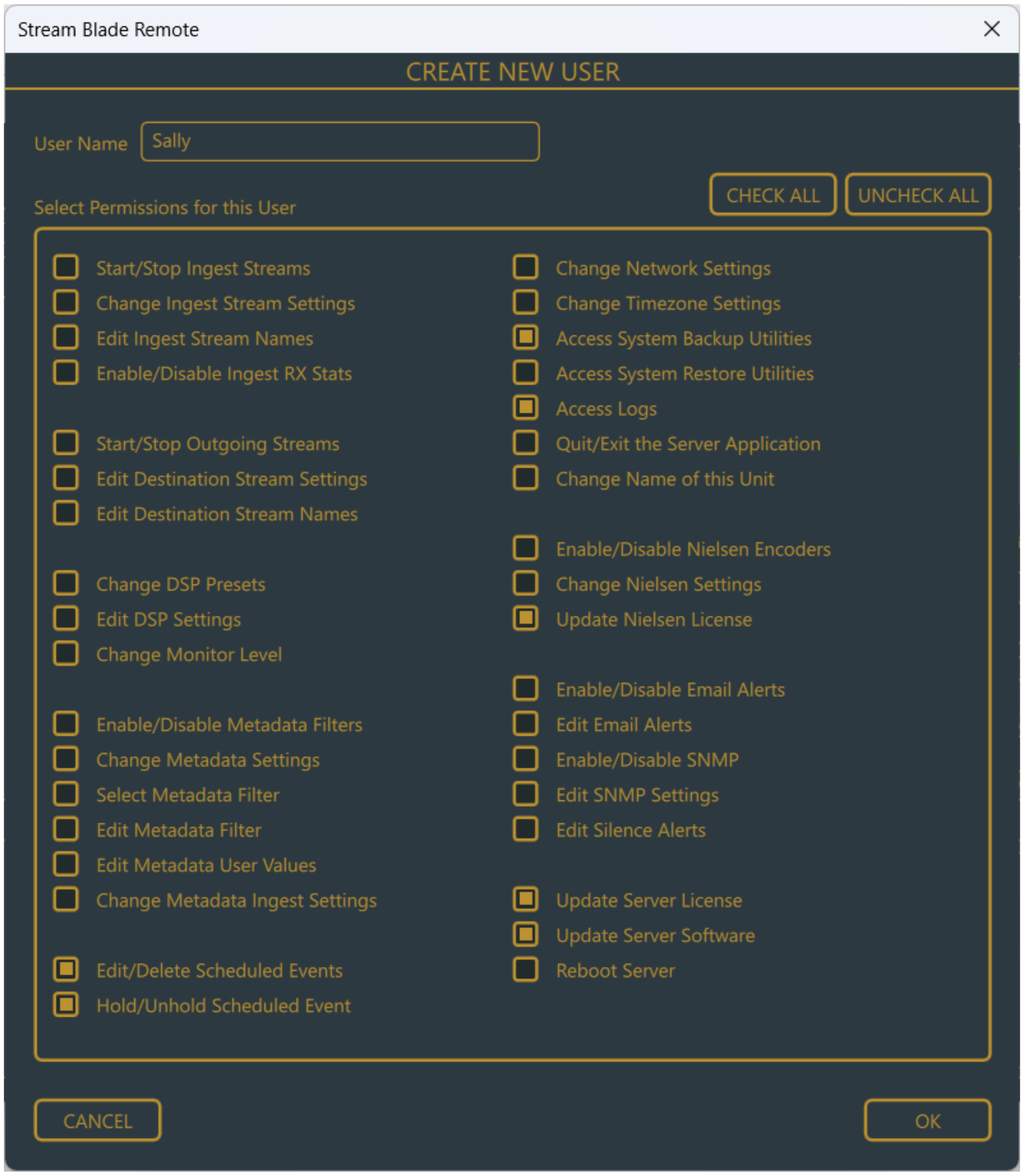
[✓ Enabled (required for CBET mode 2)] Disabled

A software license is required to use the Nielsen Watermarking feature. Contact Wheatstone Sales if you have not ordered a license and are interested in using this feature. You will also need your station's CBET information from Nielsen to set this up.

See Chapter 4 – Maintenance to learn how to apply the software license to enable Nielsen Watermarking.

USER ACCOUNTS

Wheatstream comes out of the factory with one user account, Admin, for which no password has been set -- so anyone can get in and do anything. Click the SET PASSWORD button in the top right corner to create a password for the Admin account. Once this is done, you (the Admin) may create other users with specific sets of privileges that you may choose from a list. Privileges may be revised at any time. In the example below, a new user Sally is being created with the ability to perform backups, access logs, updates licenses, and manage the scheduler.


Stream Blade Remote

CREATE NEW USER

User Name: Sally

Select Permissions for this User

CHECK ALL | UNCHECK ALL

- [ ] Start/Stop Ingest Streams
- [ ] Change Ingest Stream Settings
- [ ] Edit Ingest Stream Names
- [ ] Enable/Disable Ingest RX Stats
- [ ] Start/Stop Outgoing Streams
- [ ] Edit Destination Stream Settings
- [ ] Edit Destination Stream Names
- [ ] Change DSP Presets
- [ ] Edit DSP Settings
- [ ] Change Monitor Level
- [ ] Enable/Disable Metadata Filters
- [ ] Change Metadata Settings
- [ ] Select Metadata Filter
- [ ] Edit Metadata Filter
- [ ] Edit Metadata User Values
- [ ] Change Metadata Ingest Settings
- [✓] Edit/Delete Scheduled Events
- [✓] Hold/Unhold Scheduled Event
- [ ] Change Network Settings
- [ ] Change Timezone Settings
- [✓] Access System Backup Utilities
- [ ] Access System Restore Utilities
- [✓] Access Logs
- [ ] Quit/Exit the Server Application
- [ ] Change Name of this Unit
- [ ] Enable/Disable Nielsen Encoders
- [ ] Change Nielsen Settings
- [✓] Update Nielsen License
- [ ] Enable/Disable Email Alerts
- [ ] Edit Email Alerts
- [ ] Enable/Disable SNMP
- [ ] Edit SNMP Settings
- [ ] Edit Silence Alerts
- [✓] Update Server License
- [✓] Update Server Software
- [ ] Reboot Server

CANCEL | OK

CHAPTER 2 - WHEATSTREAM AUDIO PROCESSING

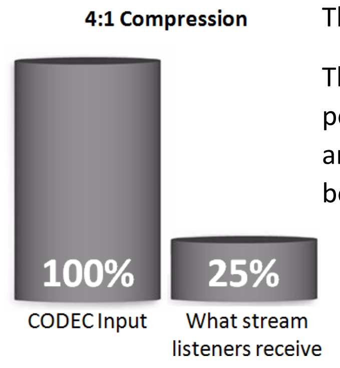
During design discussions for Wheatstone's Wheatstream product we recognized that great sounding Internet streams didn't happen by accident. It takes very carefully designed and specialized audio processing to allow codecs to deliver their best perceived audio quality. That being said, air chain processing is the most *inappropriate* processor for Internet streams, as are repurposed utility compressors and limiters. They simply don't have the intelligence for the job.

What Does A Streaming Codec Do?

The job of a streaming codec is to remove details from the audio and then hide the fact that those details have been taken away. Unfortunately, how well a codec can perform that task isn't just up to the codec; audio that is inappropriately processed can make an Internet stream sound a lot worse than it should if the codec is "making mistakes" because it can't tell processing artifacts from audio. Purpose built stream processing cleverly avoids such pitfalls and will *always* make a huge improvement in the quality of a web stream's sound.

Audio In, A Lot Less Audio Out

It's not unusual for Internet stream codecs to operate at 4:1 or higher compression. What may not be appreciated is what the seemingly low compression of 4:1 actually means. One way to think of it is "4 parts in and only 1 part out" as shown in the following example.



The left column represents the audio at the codec input.

The right column represents what's *left* of the audio after the codec has performed 4:1 compression. In other words, it's only the remaining 25% that arrives at your stream's listeners; they will never hear all of the original audio because of the missing 75%.

Fortunately, and because of how our hearing works (Google "auditory masking") humans never hear everything that's in the audio anyway and perceptual codecs capitalize on that by removing details we probably wouldn't hear, hopefully without upsetting what we *do* hear. Regardless of the amount of compression, the amount of audio data that needs to be transmitted is greatly reduced. Even a very gentle 2:1 compression removes half of the audio data.

It bears repeating that the *only* job of a perceptual codec is to remove things from the audio that most people won't notice is missing in order to make the data "fit" within the constraints of the stream's bitrate. The lower the codec's bitrate is, the more aggressively it must remove data to make the data fit.

Audio processing artifacts and other unwanted signals can "tease" a codec into making mistakes. Noise, hum, large phase errors between left and right channels, clipping distortion byproducts and non-audio signals are all undesired. Competent Internet stream processing is designed to minimize those undesirables. Again, repurposed on-air processing or utility compressors and limiters are simply not smart enough for the job.

What Makes Wheatstream Processing Different?

Taking into consideration everything in the Introduction, we then add the goals that competent Internet stream processing *must* meet to be "competent":

- Not produce, or at least minimize, all of the codec "teasers".
- Provide consistent loudness across widely varied program levels and content.
- Provide consistent spectral balance under all program conditions.
- Provide absolute peak control to keep the codec input level away from 0dBFS (there's nothing above 0dBFS but distortion, i.e. you're "out of bits!"

Wheatstone's Wheatstream accomplishes these goals using a clever combination of processing tools. All eight processing instances are completely independent:

Allpass filters: (phase rotators) make program content waveforms more symmetrical. Symmetrical audio waveforms allow audio processing algorithms to work more effectively, especially when the control of audio peaks is accomplished.

The Internet transmission medium has symmetrical positive and negative limits (like FM), therefore Wheatstream's allpass filters are always active.

A six band equalizer which includes:

- Adjustable high-pass filters to remove non-program related subsonic energy for cleaner sounding processing.
- Four parametric equalizers, two of which may be operated in shelving or parametric modes to allow tailoring of a stream's spectral balance.
- Adjustable low-pass filters to remove out of band and/or very high frequency non-program related energy. Doing so can make codecs sound better, especially when operating at very low bitrates.

A multiband main processor:

- Each band's Predictive Dynamics Controller utilizes Neural Network techniques to manage each band's gain for extremely natural management of program dynamics. Sophisticated algorithms utilize current and historical data to steer the processing to *exactly* what is required at that instant.

A highly specialized multiband final peak limiting section:

- On-air processor style Bass Management permits customizing the sound of the web stream to better match a station's on-air bass style and texture.

- Sophisticated Stereo Width Management maintains a natural stereo sound field, always preventing over enhancement of already wide stereo separation.
- Specialized multiband final peak limiters managed by Peak Energy Estimators for excellent peak control and vanishingly low inter-modulation distortion and dynamics-related artifacts.

Wheatstream's audio processing offers a wide adjustment range to allow the texture of streamed content to be customized. Anything from virtually indistinguishable from the incoming programming to wall of sound is possible. Even codecs running at very low bitrates will be presented with codec-friendly audio.

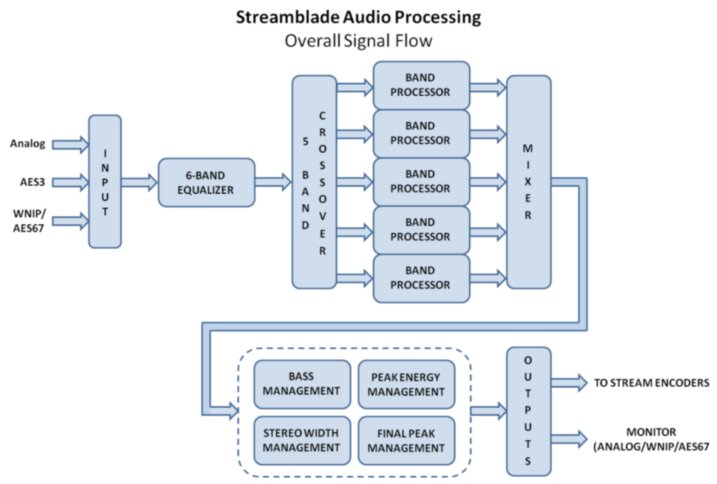
Why Process The Stream Differently Than On Air?

Mentioned previously is the possibility for Wheatstream's processed audio to sound almost identical to the incoming programming even though the dynamic range has been drastically reduced. Such flexibility was designed in for several reasons:

First, not every stream type needs to be processed by heavy handed algorithms – sometimes the goal is to hear the stream's audio sound more like the original programming. Second, the "gentler" the processing seems, the more opportunities a codec will have to remove things we won't notice. Third, being able to run "gentle" doesn't mean that Wheatstream can't also set the world on fire with radical changes in dynamic range and spectral rebalancing – if that is the stream's sound target.

Across its entire adjustable operating range Wheatstream provides consistent loudness and spectral balance; its output is *always* processed and a quick output to input comparison bears that out. Because Wheatstream's output might not "sound" processed under certain conditions, that doesn't infer that it isn't. Radical differences will always be measurable and audible in direct A/B tests.

WHEATSTREAM PROCESSING BLOCK DIAGRAM



The above diagram shows the audio signal flow for one of Wheatstream's eight audio processing instances, noting that the input can receive audio from analog, WheatNet/AES67 or AES3 sources.

StreamBlade's audio processing and Streaming Engines are hosted within its own WheatNet-IP Blade whose rear-panel analog and digital I/O adds lots of options for signal routing. Audio routing to Wheatstream's processing inputs may be modified via Wheatstone's Navigator utility.

Each processing instance has its own Input Gain control with a +/-12dB available adjustment range to best match the incoming program levels. The default Input Gain setting is 0dB in all Factory Presets which presents a unity gain (0dBFS) signal path to the outside world.

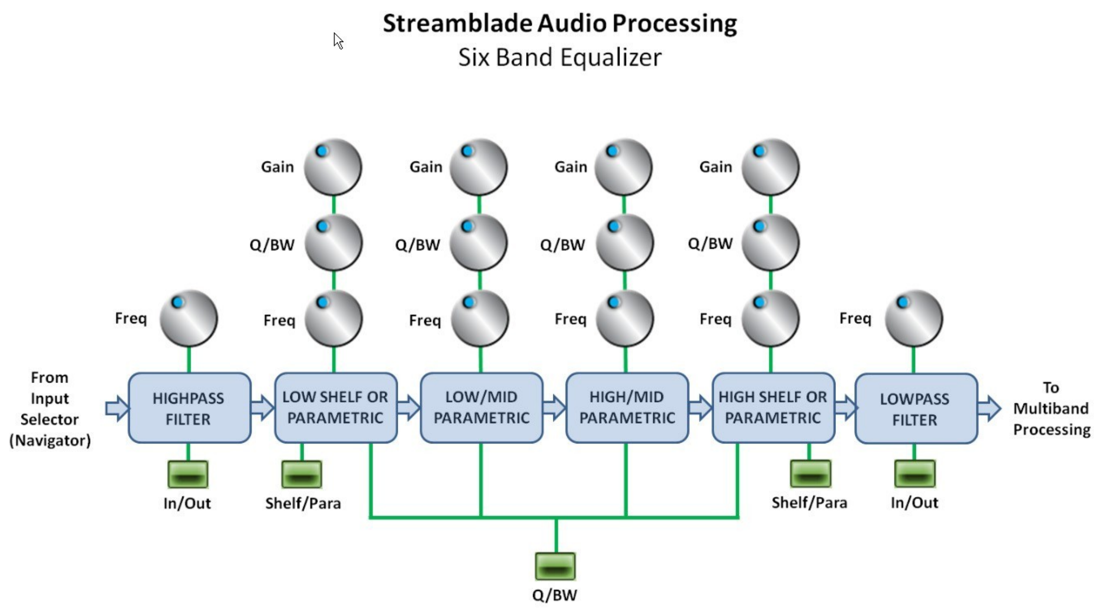
Each of Wheatstream's <u>processing</u> instances can feed *five* different destinations:

- Four Streaming Engines dedicated to HLS, ICECAST, RTMP and RTP destinations.
- Audio outputs for pre-codec monitoring to allow Wheatstream's processing to be monitored in real time rather than waiting for the audio stream to be decoded.
- The processed audio being fed to the Streaming Engines and the monitor outputs share a *common* Output Level control with an adjustment range of -24dB to +0dB. Setting the Output Level control to 0dB permits the maximum permissible audio levels into the codec; by design about -0.25dBFS when all processing is enabled and Wheatstream is running a Factory preset.

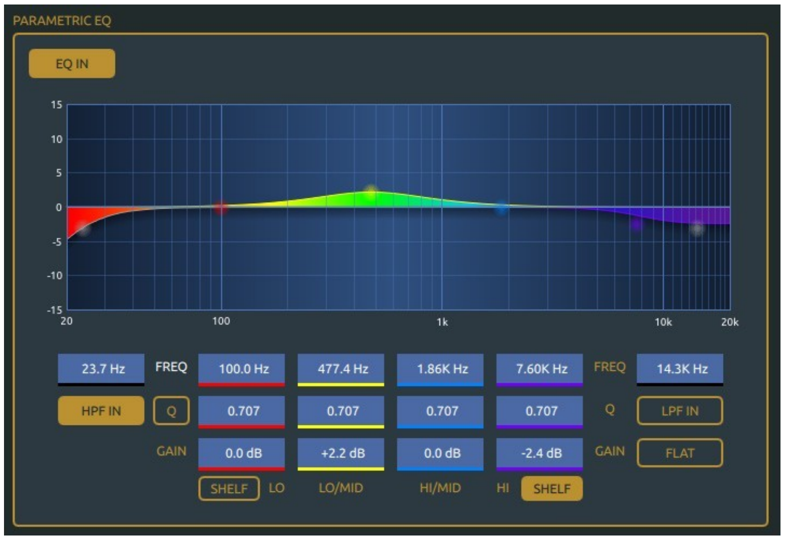
As noted above, the gain structure of Wheatstream's final audio limiters is configured to achieve a maximum peak level of -0.25dBFS into the Streaming Engines when the Output control is set full scale to 0dB. Constraining the maximum peak output level to below digital full scale prevents unintentional downstream headroom issues which might otherwise occur if gain were allowed to be added *after* Wheatstream's limiters.

If the final limiter is switched Off or is operating with long attack times peak output levels may be indeterminate and may try to exceed 0dBFS. This will result in distorted stream audio.

SIX BAND EQUALIZER



The signal path through Wheatstream's equalizer section is shown above, including the high pass and low pass filters and the four parametric equalizers which may operate in parametric or shelving mode.



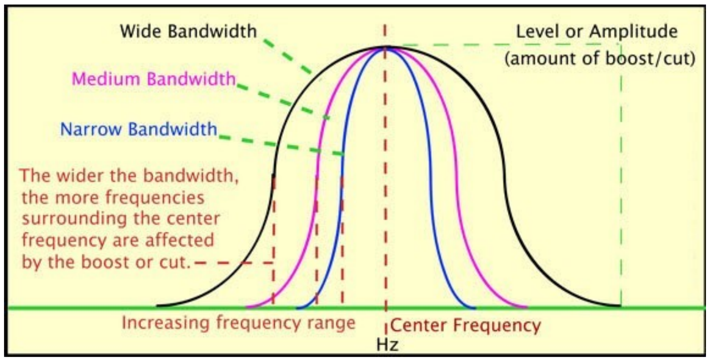
At left is a screenshot of Wheatstream's equalizer controls. At upper left is the EQ IN/EQ OUT button which will illuminate when the equalizer is active.

The HPF IN button at lower left enables the high pass filter which removes unwanted low frequency energy. Above the HPF IN button is the filter's frequency selector.

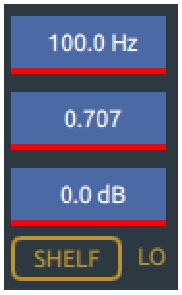
The equalizer's Q/BW button selects whether the parametric equalizer will use Q or Bandwidth parameters for setting how equalization affects adjacent frequencies. Higher Q settings create a narrower filter as do lower bandwidth settings so functionally, Q and Bandwidth could be considered reciprocal.

The following compares Bandwidth, Q, and the amount of audio spectrum affected by the equalizer *around* a selected center frequency.

BW (Octaves)	Q
2.0	0.667
1.0	1.414
2/3 (0.66666)	2.145
1/2 (0.50000)	2.871
1/3 (0.33333)	4.318
1/6 (0.16666)	8.651
1/10 (0.10000)	14.242
1/30 (0.03000)	43.280


Wide Bandwidth
Medium Bandwidth
Narrow Bandwidth
Level or Amplitude (amount of boost/cut)
The wider the bandwidth, the more frequencies surrounding the center frequency are affected by the boost or cut.
Increasing frequency range
Center Frequency
Hz

There are four parametric equalizer sections. Sections one and four may be operated in either shelf <u>or</u> parametric mode.



FREQ: Adjust the frequency where boost or cut will take effect.

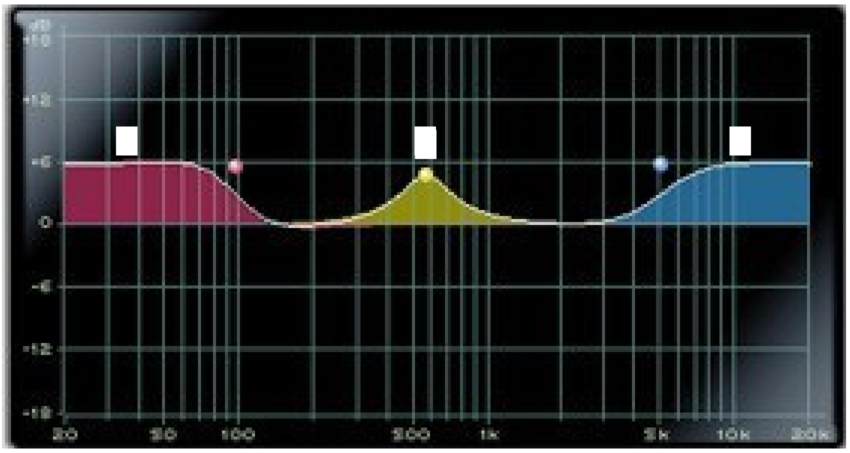
Q/BW: Sets how much audio spectrum around FREQ is affected by boost or cut.

GAIN: Sets the boost or cut applied to the selected FREQ.

SHELF: Selects equalizer band one's Shelf mode (rather than parametric).

The Shelf and Parametric modes shape the audio spectrum in different ways. Shelf mode affects all frequencies <u>above or below</u> a particular frequency while parametric mode affects frequencies <u>either side of a center frequency</u>. The graphic below compares the equalizer behavior when in shelf or parametric modes.

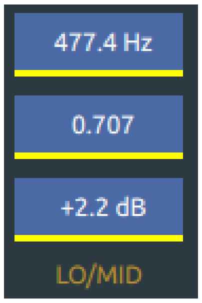
Shelf Parametric Shelf



A Shelf response is useful for boosting overall bass heft or brightness.

A Parametric response is useful for enhancing or reducing a narrow band of energy around a chosen center frequency.

The LO/MID and HIGH/MID equalizers have only parametric behavior and are useful for boosting or cutting just a *specific* band of frequencies around a desired *center* frequency. There are three controls associated with the LO/MID and HIGH/MID equalizers:



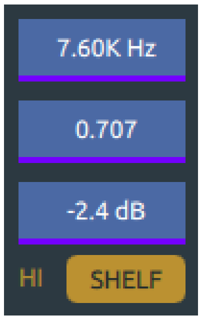
FREQ: Adjusts the frequency where boost or cut will take effect.

Q/ BW: Sets how much audio spectrum around FREQ is affected by boost or cut.

GAIN: Sets the boost or cut applied to the selected FREQ.

Note that the parametric equalizers may be *overlapped* to create highly specialized filters.

The fourth equalizer may be operated in either its shelf or parametric mode.



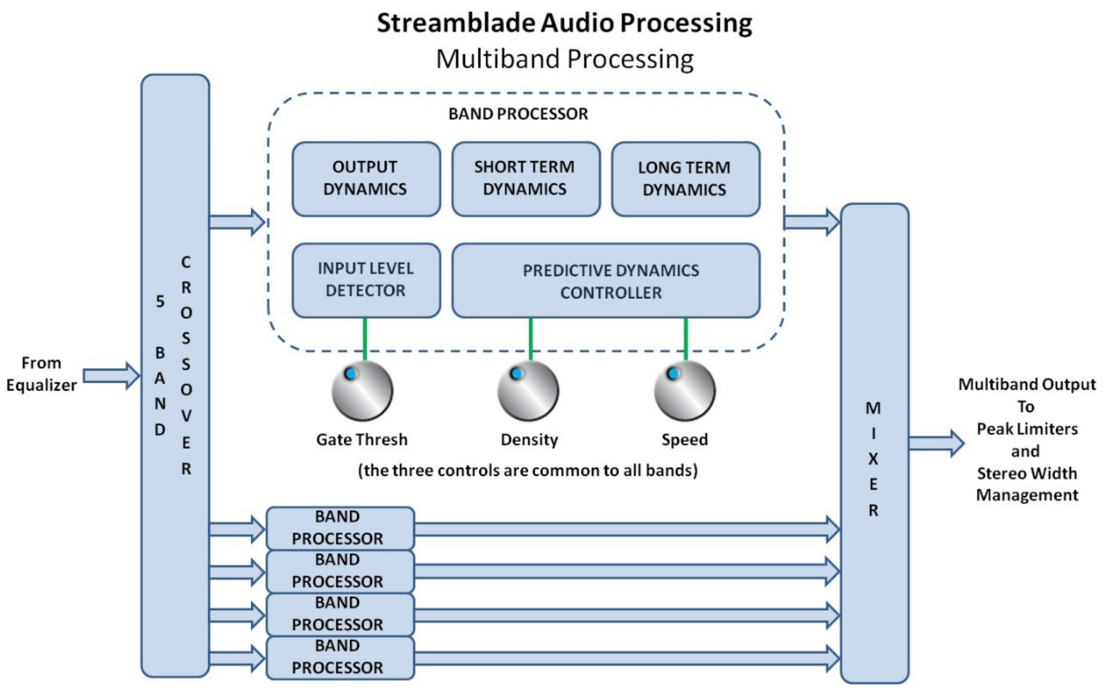
FREQ: Adjusts the frequency where boost or cut will take effect.

Q/BW: Sets how much spectrum around FREQ is affected by boost or cut.

GAIN: Sets the boost or cut applied to the selected FREQ.

SHELF: Selects equalizer band four's Shelf mode (rather than parametric).

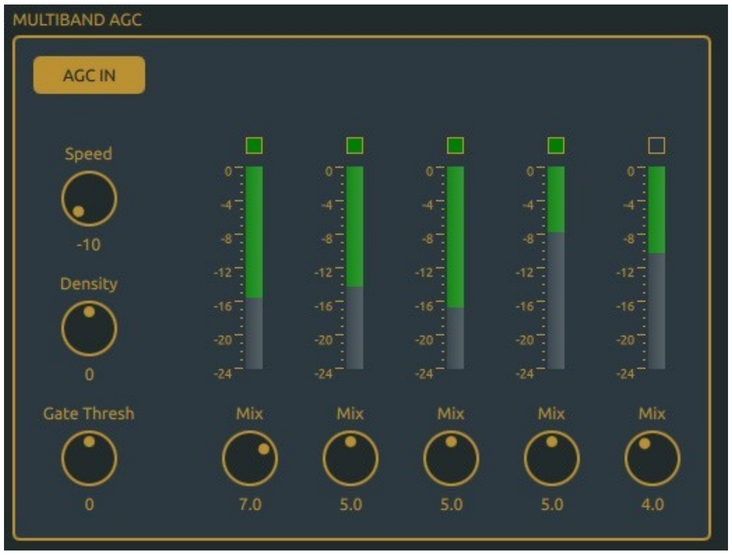
MULTIBAND PROCESSING SECTION



The above diagram shows the signal flow through the five-band processor with extra detail shown for Band One.

AGC USER CONTROLS

Each of the five processing bands has a gain reduction meter with a zero to -24dB scale which shows the depth of processing at that moment.



Located directly above each gain reduction meter is the GATE indicator for the band. The GATE indicator will be ON (lit) when the input level to a band is *above* the setting of the Gate Thresh control. In other words, a lighted Gate indicator shows that a band is actively engaged.

Under certain processing conditions one or more gain reduction meters may appear to be at full scale – maximum gain reduction – however this is no cause for concern. There is *plenty* of processing headroom beyond a meter's -24dB indication, but unlike conventional processing Wheatstream's audio does not get more dense or "tighter" as the gain reduction gets deeper, even if pushed well beyond -24dB full scale.

The user controls Gate Thresh, Density, and Speed are common to all five bands. Though they *are* common controls how they affect the processing in each band on a moment to moment basis depends on past and present characteristics of the unprocessed *and* processed audio within that band.

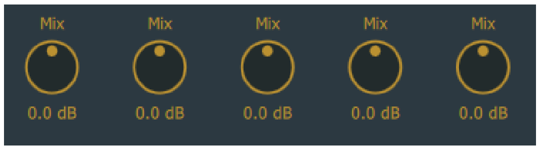
SPEED**:** Like the other two "processing" controls the Speed control has 21 positions from -10 to +10. 0 is the default setting in all Factory presets. Higher settings cause processing to more tightly regulate a band's output level while lower settings have the opposite effect. Unlike conventional processing, Wheatstream's Speed control does *not* affect release times but only increases *opportunities* for a band to change its gain.

DENSITY**:** While Wheatstream processing can invisibly manage the audio levels of all program types, sometimes invisible is not what is wanted. The Density control allows the user to dial in some "excitement" or "radio show" for how Wheatstream sounds as it processes the audio.

GATE THRESH**:** The Gate Thresh control determines a band's threshold upon which gain changes are frozen if the input audio level falls. When a band is gated its gain is held steady so as to not increase noise. Lower Gate Thresh settings permit the input level to fall farther before freezing the gain. Higher settings increase the threshold for when a band's gain will unfreeze.

Referring to the block diagram, each band's Input Level Detector, Output Dynamics, Short and Long Term Dynamics and Predictive Dynamics blocks measure certain characteristics of the incoming and processed output audio. Such data allows each band's Neural Network to "steer" its attack times, release times, compression ratios and control slopes to ensure that the band's output is exactly what the user desires given his settings of the Speed and Density controls.

Of note is the complete absence of the typical multiband coupling controls. Wheatstream's intelligence ensures that the bands never wander too far from each other, even though the multiband Linkwitz-Riley crossover slopes are rather steep.



MIX: Each band of processing has its own output Mix control to allow the spectral balance of the processed audio to be customized. The Mix controls have a +/-6dB adjustment range with 0.1dB steps. In Factory presets the Mix controls will be in their 0dB positions.

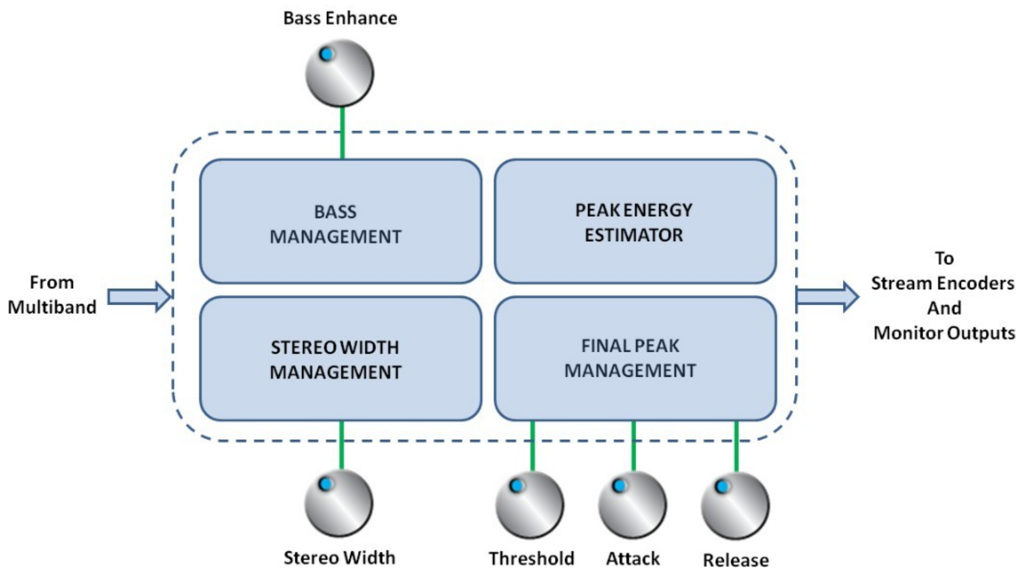
Another role of the Mix controls is setting how hard the Final Limiter section is driven. As a Band Mix control is turned clockwise its output level increases which then drives the Final Limiters harder and causes them to do more limiting. It follows that a generic increase in overall limiter drive can be accomplished by advancing <u>all five</u> Mix controls clockwise by an equal amount. Likewise, a decrease in final limiter drive may be accomplished by turning the Mix controls in the opposite direction. Keep this behavior in mind if the drive to the final limiter needs to be adjusted.

FINAL LIMITING

The final Limiter has its own IN button which is illuminated when that section is active. The gain structures of the multiband section and back end limiters are arranged so that the *average* output level doesn't radically change when sections are switched in and out. The peak output level *will* increase however when the limiters are bypassed due to the lack of peak limiting.

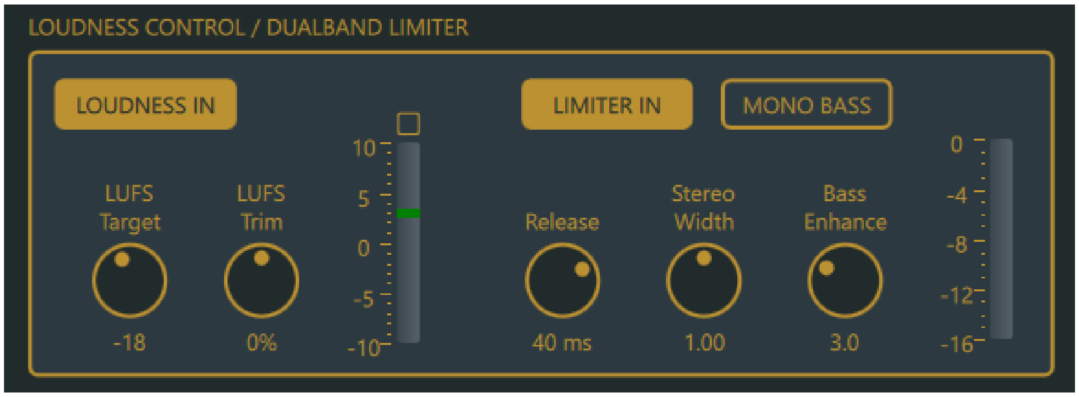
Wheatstream's Dual Band Limiting is where the majority of "codec-friendly" processing occurs. In very general terms the final limiting separately occurs in two bands, above and below 250Hz.

Streamblade Audio Processing
Final Limiting

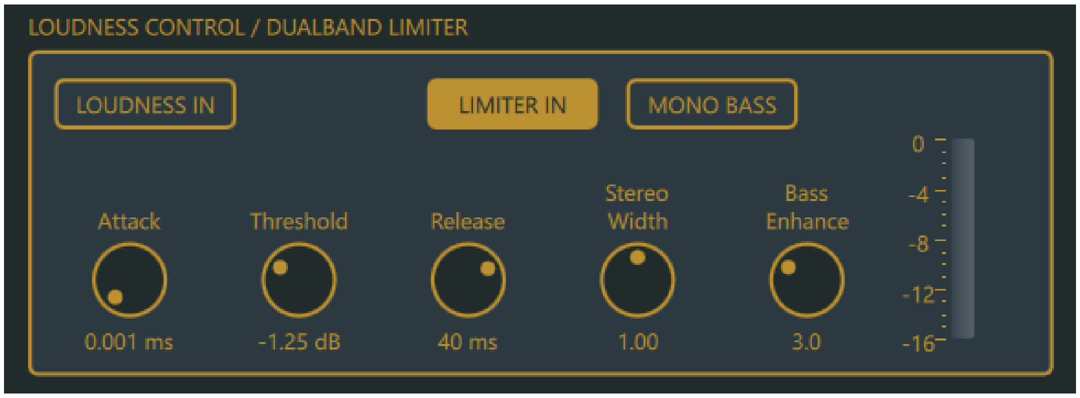


LIMITER USER CONTROLS

The Dual Band Limiter offers a different subset of controls to the user depending on whether the Loudness control is enabled or not. When it is enabled, you have controls for LUFS target and trim, as well as a LUFS meter, while knobs for adjusting Attack and Threshold have been removed since these parameters are now under control of the loudness algorithm:



When loudness control is disabled, the LUFS controls are replaced with knobs for adjusting Attack and Threshold:



Limiter In/Out

When Limiter In is illuminated the Dual Band limiter sections are enabled; all related processing is active. If the Limiter In button is not illuminated peak control is disabled and the peak audio output level is uncontrolled and therefore could be high enough to cause audio clipping in the stream encoder.

Mono Bass

To assist low bitrate codecs in their endeavor to reduce the data to be streamed it can be beneficial to remove low frequency energy in the stereo difference channel (L-R). For Wheatstream that includes frequencies below 250Hz.

The "Mono Bass" option is effective because our ears are less sensitive to stereo separation at low frequencies, a fact that makes stereo separation redundant for a good perception of stereo. When the Mono Bass button is illuminated Wheatstream removes energy below 250Hz in the L-R stereo difference channel (only!). With such energy removed the codec doesn't see or try to encode it, leaving more bits available for encoding things which are more perceptible.

Attack

Sets the attack time of the final limiters in Wheatstream's limiting section. To accommodate the widest possible use cases the adjustment range has been made quite large: 10 microseconds to 1 millisecond (0.001 to 1.0mS). Short attack times below about 50 microseconds (0.005) will provide extremely tight limiting with virtually zero overshoots, and without resorting to clipping.

In all Factory Presets the Final Limiter Attack time is set quite fast, the Limiter Threshold is set to -0.25 dBFS, and the Output Gain control is set to -3.0dB. With these settings the processor's final peak level output should be no higher than -3.25dBFS, approximately the industry recommended maximum peak audio input level for streaming codecs.

Release

Sets the release time of the final limiters in Wheatstream's limiting section. Like the attack time, the release time adjustment range has been made quite broad: 1.0 millisecond to 100 milliseconds. Although wide, this adjustment range was provided both for "artistic" reasons and to accommodate all possible use cases.

Note that setting the limiter release times below about 10 milliseconds may create some artifacts with sustained limiting depth beyond a few dB. The most effective peak control occurs when the limiter is doing no more than about 3dB on peaks. Deeper and/or more sustained limiting provides no real benefit and can create listener fatigue.

Threshold

Sets the threshold of limiting in dBFS (decibels below digital full scale) for the final limiters which then determines the maximum peak level fed to the Output Level control.

Stereo Width

In streaming, the stereo image width needs to be managed in order to optimize how codecs sound, especially those operating at low bitrates. Preventing excessive stereo image width helps mitigate the "codec swishies" and other artifacts not part of the original program audio.

With the Stereo Width control set to 1.0, as it is in Factory presets, the encoded stereo image width will be very similar to the original source material and will remain so over a very large range of program content.

Stereo Width settings lower than 1.0 can be advantageous for codecs operating at or below 64kbit/s. Likewise, codecs operating at or above 128kbit/s can *usually* encode additional L-R stereo information without generating undesirable artifacts. The range of the Stereo Width control has been constrained to prevent codec-teasing and excessive stereo image width. When the control is set to zero, energy above 250Hz will be reduced to mono *(see below). When the control is set to its 2.0 position the stereo image width above 250Hz is enhanced by 6dB.

***Special Note**: If the Stereo Width control is advanced to zero and the Mono Bass function is not enabled, stereo separation will only exist below 250Hz unless/until the Mono Bass function is activated.

Bass Enhance
The Final Limiter is equipped with an on-air style Bass Processor to allow better matching of a station's Internet stream to its on-air sound. The Bass Enhance algorithm operates only on L+R (mono) frequencies below 250Hz, and unlike the Mono Bass function has minimal effect on the sound of low bit rate codecs, even with generous bass enhancement dialed in.

Output
Sets the peak audio output level in dBFS (deciBels Full Scale), of each processing instance and therefore the input to its Streaming Engine. It also sets the output level of the monitor outputs and the processed audio available within the WheatNet-IP domain.

The Output Gain control has an adjustment range of -12dBFS to 0dBFS in 1dB steps and is set to -3dBFS in Factory presets.

LOUDNESS CONTROLLER
Wheatstream's Loudness Controller has been designed to be tightly compliant with the latest BS.1770 and R128 standards.

The controller has two operating controls, LUFS Target and LUFS Trim, along with a bargraph meter which shows the controller's activity while managing program loudness as the incoming source material changes.

Embedded within the loudness controller algorithms are special weighting functions to make the loudness controller as unobtrusive as possible, regardless of the incoming program content.

LUFS Target
This control sets the medium and long term loudness over a range of -10 LUFS to -24 LUFS and in one dB steps. Whenever the LUFS Target control is adjusted to a new loudness setting, new time constants temporarily come into play to quickly set the loudness to the new level regardless of whether the new level is higher or lower than the previous one.

LUFS Trim
The LUFS Trim control acts as an offset for the LUFS Target control to allow the user to make slight changes (+/-10%) to the actual LUFS Threshold. This control is useful to very precisely 'dial in' the desired loudness to more exactly match the LUFS Threshold whenever previous processing adjustments have been made that cause the actual loudness to be somewhat offset from the setting of the LUFS Target control.

Bargraph Meter
To the right of the LUFS Trim control is a vertical meter with a -10 to +10 scale which is relative only and not calibrated in dB or any other parameter. During normal loudness controller operation the meter will hover somewhere near center scale and will show whether the controller is increasing or decreasing loudness based on the incoming program content. Very loud or very soft program content may cause the meter to indicate near its extremes but this is no cause for alarm.

Gate Indicator

Above the vertical meter is an indicator which shows whether the loudness controller's gain is actively following the incoming program or has ceased following it. When the indicator is illuminated it means that the incoming audio from the preceding audio processing has either stopped altogether, or the level has fallen so low as to be outside the capture range of the loudness controller. When in this state the controller's gain is frozen at its last value and will remain at that value unless or until the incoming audio is again back within the controller's capture range. Note that in this state it is impossible for the loudness exiting the controller to exceed the currently set LUFS Target because the audio level is simply too low.

DESIGN PHILOSOPHY

Users operating both a BS.1770 and CBS loudness measuring devices (such as the one in the free Orban Loudness Meter) may be surprised to learn that unlike other products performing loudness control, Wheatstream's loudness controller satisfies both the BS.1770 and CBS measurement schemes.

The reason for this is purely subjective, but we felt important, in our own implementation of a compliant loudness controller. During development of our own, and during comparison with other loudness controllers, we discovered that the audio sounded much less constricted and had far more natural dynamics when our loudness controller also made the CBS loudness meter 'happy'. Given that our company's mission has always been to make audio sound great, we decided that our loudness controller implementation needed to do two things instead of just one: first, make loudness control tightly compliant to the BS.177 and R128 stanrdards and second, sound as natural as possible by leaving important but subtle dynamics clues intact.

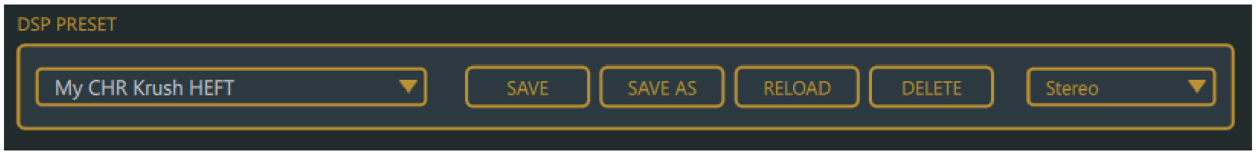
A FEW WORDS ABOUT WHEATSTREAM FACTORY PRESETS

Wheatstream comes equipped with a generous selection of Factory Presets with around 50 presets being the norm. The intent of this large a selection is to permit an end user to deeply explore the wide variety of dynamics and tonal textures that Wheatstream can create.

The multiband processing section's "Speed" and "Density" controls allow each preset to be highly modified from its original Factory settings. As a data point, modifying 50 Factory presets using just those two controls can create over 22,000 combinations of sound and texture.

When the five Band Mix controls and their adjustment ranges are considered, there are even more choices; in fact over 13 million different combinations of sound, texture and spectral balance are available with those 50 Factory presets!

As is standard fare on most modern audio processors, Wheatstream's Factory presets are configured so they cannot intentionally be overwritten, creating a safe place to return to if one happens to get "lost".



The Preset dialog's SAVE, SAVE AS, RELOAD and DELETE buttons do exactly as their labels suggest, though the RELOAD button might need a brief explanation.

If a Factory Preset has been recalled and then modified by the user controls, clicking the RELOAD button simply reloads the last taken preset without having to go back to the list of presets to locate and select it again. Consider the RELOAD button a shortcut to resetting the preset parameters back to the Factory settings before they were modified by the user controls.

WE APPRECIATE AND DESIRE USER FEEDBACK FROM THE FIELD!

If you find that the factory presets don't result in the sound you are seeking, please reach out! We can be reached via our Website's Tech Support Portal, as well as by email or phone:

On the web:	www.wheatstone.com/support
Via email:	techsupport@wheatstone.com
Via telephone:	(252) 638-7000

CHAPTER 3 – METADATA: A PRIMER

WHAT IS METADATA?

Metadata is the term used to describe textual information that goes along with an audio stream – things like artist, title, album, duration, ISRC, URL, etc. It applies not only to songs, but also to ads, sweepers, liners, station IDs, PSAs – the role of metadata is to describe and distinguish all of these event types. Metadata is typically generated by playout systems (automation systems) and forwarded to the stream encoder for further processing. In other cases the metadata is sent to a validation or "cleansing" service en route to the streaming encoder. (For simplicity in the following discussion, we'll refer to any source of metadata as the "automation system") Metadata contains the "now playing" data that end-users (listeners) see displayed on their player apps.

The first thing to understand about metadata is that there are no industry-wide standards that govern its structure, content, or format. The second thing to understand is that every CDN which ingests metadata has its own unique recommendations and/or requirements for the structuring and formatting of that data, else it will not be accepted. Add to this the fact that transport protocols (e.g. HLS/ICY/RTMP) impose their own constraints on how metadata is carried. Thus, from a high-level systems view, we have a data-processing task in which the input may arrive in any format, which then needs to be transformed into some other format, the exact details of which cannot be known until it is necessary to carry out this task.

It is not a hopeless task, however. There may not be industry standards for transmitting metadata, but there are common practices. Many automation systems export (transmit) metadata in XML or quasi-XML format, like these examples:

Ex.1

```
<nowplaying>
<sched_time>53592600</sched_time>
<air_time>54235000</air_time>
<stack_pos></stack_pos>
<title>Redneck Crazy (Radio Edit</title>
<artist>Tyler Farr</artist>
<trivia>Note</trivia>
<category>M01</category>
<cart>N825</cart>
<intro>13000</intro>
<end></end>
<station>K95DEMO</station>
<duration>208900</duration>
<media_type>SONG</media_type>
<milliseconds_left></milliseconds_left>
<Album>Redneck Crazy</Album>
<Field1></Field1>
<Field2></Field2>
<Label>Columbia</Label>
<YearSG>2013 SG</YearSG>
</nowplaying>
```

Ex.2

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
```

```
<audio ID="id_3155460704_30756200">
<type>Song</type>
<status>Playing</status>
<artist>ALAN JACKSON &amp; JIMMY BUFFETT</artist>
<title>IT'S FIVE O'CLOCK SOMEWHERE</title>
<number>837292</number>
<length>00:03:49</length>
</audio>
```

Another popular format for conveying metadata is in key=value pairs, like this one:

Ex. 3

```
cat=Song artist=Tom%20Cochrane title=Life%20Is%20A%20Highway album=Mad%20Mad%20World
duration=04%3A25
```

Note that in this particular example, spaces are used specifically to separate key/value pairs, which means that spaces which occur in the value segment must be conveyed in some other manner. In this case they are "percent-encoded," which is to say, replaced with the hexadecimal ASCII value of the space character (20), signaled with the percent character. Note that the colon in the duration value of 04:25 is also percent-encoded: %3A

Other details in the examples above are worth pointing out. (A) In Ex.1 the duration value is specified in *milliseconds*; in Ex.2 it is called "length" and specified in *hh:mm:ss* format; and in Ex.3 it is again called "duration" but specified in percent-encoded *mm:ss* format. (B) We can determine the type of event from the *<media_type>* tag in Ex.1, from the *<type>* tag in Ex.2, and the *cat*= key in Ex.3. (C) The *artist* value in Ex.2 contains an HTML "entity" – the *&amp;* sequence which represents the ampersand character in HTML documents. In many cases HTML entities must be converted to a Unicode value before transmission to the CDN. (D) The *title* value in Ex. 1 has a typo (or omission in this case) that we can do nothing about: the closing parenthesis in "(radio edit)" is missing. [Actually one could do something about it – see discussion of Lua transform filters below.]

The examples above are all for song events, but automation systems transmit metadata for other types of events as well:

Ex. 4 – Top of the Hour Station ID

```
<nowplaying>
<sched_time>0</sched_time>
<air_time>171000</air_time>
<stack_pos></stack_pos>
<title>Top of Hour</title>
<artist>Standard TOH</artist>
<trivia></trivia>
<category>ID5</category>
<cart>BIAA</cart>
<intro>0</intro>
<end></end>
<station>95.5HD2</station>
<duration>12600</duration>
<media_type>UNSPECIFIED</media_type>
<milliseconds_left>12576</milliseconds_left>
<Album></Album>
<Field2></Field2>
<ISRC></ISRC>
<Label></Label>
<Tempo></Tempo>
<YearSG></YearSG>
```

```
</nowplaying>
```

Ex. 5 – A 29.8-second PSA treated as an ad (spot)

```
<nowplaying>
<sched_time>2429400</sched_time>
<air_time>2809000</air_time>
<stack_pos></stack_pos>
<title>3 All Stations PSA/Dept Vet Affairs</title>
<artist>VBARADPSA002</artist>
<trivia>choice, choose V-A</trivia>
<category>COM</category>
<cart>9009</cart>
<intro>0</intro>
<end></end>
<station>95.5HD2</station>
<duration>29800</duration>
<media_type>SPOT</media_type>
<milliseconds_left></milliseconds_left>
<Album></Album>
<Field2></Field2>
<ISRC></ISRC>
<Label></Label>
<Tempo></Tempo>
<YearSG></YearSG>
</nowplaying>
```

Ex. 6 – A 30-second ad

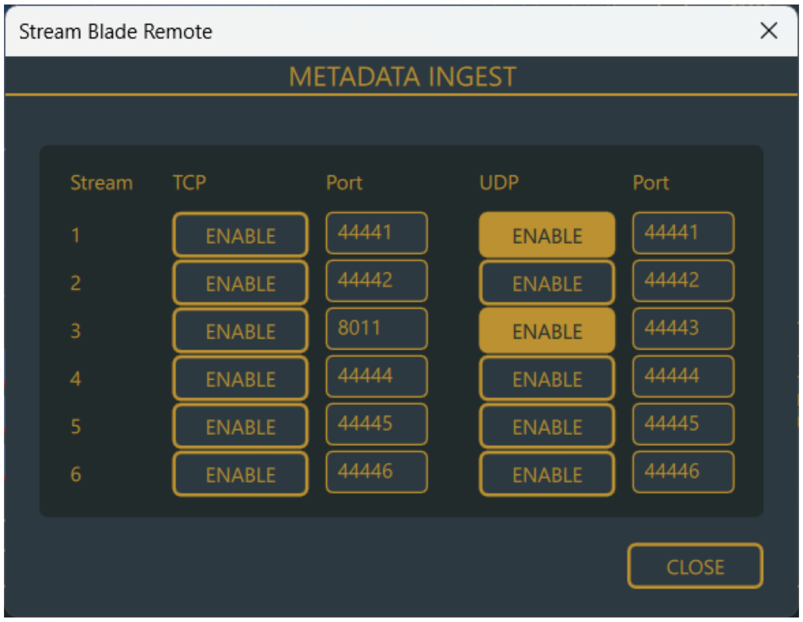
```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<audio>
<type>Spot</type>
<status>Playing</status>
<artist>WW_QRPI04660000</artist>
<title>PROGRESSIVE INSURANCE</title>
<number>29669</number>
<length>00:00:30</length>
</audio>
```

Ex. 7 – A 14-second background bed for a voice track

```
type=Link artist= title=VT%20Background album= duration=00%3A14
```

INGESTING METADATA

Wheatstone Streaming Encoders ingest raw metadata via TCP or UDP, on any port. Most automation systems have the capability of sending metadata via these channels. Click the METADATA INGEST button in the main menu pane to open the ingest dialog:


Stream Blade Remote

METADATA INGEST

Stream	TCP	Port	UDP	Port
1	ENABLE	44441	[✓ ENABLE]	44441
2	ENABLE	44442	ENABLE	44442
3	ENABLE	8011	[✓ ENABLE]	44443
4	ENABLE	44444	ENABLE	44444
5	ENABLE	44445	ENABLE	44445
6	ENABLE	44446	ENABLE	44446

CLOSE

When a metadata port is enabled, any metadata received on that port is displayed in the main Status Pane. When configuring your Wheatstone Streaming Encoder for metadata processing, this is the first step – to familiarize yourself with the nature of the incoming metadata. There is a wide choice in playout systems and other sources of metadata, and since most of them provide extensive options for formatting and customizing metadata, the metadata that you see is likely to be unique to your facility. Note that the PAUSE button at the top right of the Status Pane is useful for freezing the display in order to closely examine the received metadata.

FORWARDING METADATA

As of this writing, no CDN that we are aware of accepts metadata *in exactly the same format* as received from the automation system. It must in all cases be transformed in some manner, determined both by the transport protocol and by the CDN.

The CDN determines what kind of metadata they want. *Artist*, *title* and *duration* and pretty are much universal. Many CDNs accept *album*, *label* and *ISRC* data, using that information to provide additional features to the listener, such as album art, lyrics and clickable URLs, as well as for royalty tracking.

The event *type* is needed as a signal for geographically-based ad replacements, a service provided by most CDNs today. Events are usually grouped into 3 categories or types: songs (any music, or more generally, the program content itself, which could for example be talk rather than music); commercials, usually signaled as SPOT or COM; and everything else. Ad-insertions are triggered with the receipt of a SPOT event, and cancelled with receipt of any other event type.

HLS

HTTP Live Streaming was developed by Apple Computer and made available to all users and vendors. Technically it is not really a stream; rather it consists of a series of “web pages” (called segment files), each of which contains a few seconds of compressed audio data, which are continuously downloaded by the player in the same manner that any browser downloads a web page. The stream URL actually points to a “manifest file,” which is just a text file containing pointers to the individual segment files within the current window. As the window moves forward in time with the creation of each new segment file, the manifest file is also updated, dropping the oldest segment and adding the newest.

In HLS streams, metadata is primarily conveyed by means of ID3v2 frames interjected between AAC frames, which keeps the metadata synchronized with the audio stream with a precision of about 20 milliseconds. Each HLS segment is also prefixed with the most recent ID3v2 frame, so that players connecting to the stream between metadata events are nonetheless updated immediately.

ID3v2 frames may carry common metadata such as artist, title, album, duration, ISRC and URL. They may also carry custom data of any sort, a capability which may be used to trigger ad insertions.

Ad-insertions may also be triggered by means of SCTE-35 switching directives included in the manifest file.

ICY

Icecast is an open-source streaming protocol which has proliferated widely, because it is easy to use and was one of the first streaming protocols introduced. Many variations on the original server architecture exist, which has led to an equal number of variations in the structure of metadata updates.

Metadata is conveyed to the Icecast server as *out-of-band* data -- that is, it is neither part of, nor embedded into, the audio stream, but rather takes its own route over the public internet to the server, without benefit of timestamps or any other synchronization mechanism. Thus it can never be precisely timed. [Some CDNs have developed an in-band method for metadata transmission over an Icecast stream, but the protocol only conveys a small portion of the available data, and has not been widely adopted. Wheatstone streaming encoders do not currently support the in-band method.]

The canonic Icecast metadata update message is in this form, with separate "query fields" for artist, title, duration, etc:

```
http://username:password@myicy.server.com/admin/metadata?mount=/MountName&mode=updinfo
&title=Some%20Song&artist=Some%20Band&duration=00:04:13&type=SONG
```

But there are many variations. One common pattern is to put all of the data into a single *&song=* query field (even for non-song events), which requires the use of a special separator character (typically the vertical bar) to delineate the constituent elements. This requires that the elements in the metadata be conveyed in a specific order, and that any missing element be represented nonetheless by the presence of a separator character. The order of the components is determined by the CDN. In the following examples the order is: artist | title | duration | type.

```
http://username:password@myicy.server.com/admin/metadata?mount=/MountName&mode=updinfo
&song=Some%20Band|Some%20Song|00:04:13|SONG
```

```
http://username:password@myicy.server.com/admin/metadata?mount=/MountName&mode=updinfo
&song=|PROGRESSIVE%20INSURANCE|00:00:30|SPOT
```

```
http://username:password@myicy.server.com/admin/metadata?mount=/MountName&mode=updinfo
&song=||00:00:12|OTHER
```

MRV2

Media Relay Version 2 is the proprietary streaming protocol developed by Triton Digital to provide a high degree of resilience to errors and packet loss, built-in redundancy, and accurate synchronization of metadata with the audio stream. Metadata values are collected from the incoming event messages and arranged into a string of simple key=value pairs, which is then passed to the MRV2 manager for injection into the audio stream.

RTMP

The Real-Time Messaging Protocol was developed by Adobe (Macromedia) originally for transport of Flash video. Even though Flash is no longer actively supported, RTMP itself is still widely used to transport audio data of any format, although these days probably more often for backhaul up to a CDN or distribution center, than for listener player streams.

Metadata may be embedded into the audio stream by means of so-called “setDataFrame” packets. These packets have a particular format which involves setting values for three different “properties,” which are Title, Artist and URL. In practice, the URL field is a catch-all which contains all the metadata which is neither title nor artist. It must also be formatted as an actual URL which could be transmitted via HTTP.

To see how this works, consider the following event received from the automation system:

```
<nowplaying><sched_time>220200</sched_time><air_time>402000</air_time><stack_pos></sta
ck_pos><title>The Sky Is A Neighborhood</title><artist>Foo
Fighters</artist><trivia>*</trivia><category>MNJ</category><cart>R585</cart><intro>140
00</intro><end></end><station>93.5HD1</station><duration>243300</duration><media_type>
SONG</media_type><milliseconds_left></milliseconds_left><Album>Concrete and
Gold</Album><Field2></Field2><ISRC>USRW9170028</ISRC><Label>RCA</Label><Tempo></Tempo>
<Year>2017</Year></nowplaying>
```

This would be translated into RTMP format roughly like this, ignoring the various internal header and length bits that are part of the Action Message Format (AMF) used for transmitting text in RTMP streams:

```
@setDataFrame
onMetaData
title:The Sky Is A Neighborhood
artist:Foo Fighters
url:http://www.blaze105.com?autoID=R585&autoCat=SONG&cat=MNJ&duration=243&album=Concre
te%20and%20Gold&label=RCA&ISRC=USRW9170028
```

The choices and names of the query parameters in the URL string are determined by the CDN. Note that a few of the tags in the original metadata event are renamed according to the CDN’s needs. The `<cart>` tag becomes `autoID`, the `<media_type>` tag becomes `autoCat`, the `<category>` tag is reduced to `cat`, and `<duration>` is rounded down to a whole number of seconds. For this particular CDN, the `autoCat` value is used for signaling the beginnings and endings of ad-insertions. Note that the ISRC code received from the automation machine is invalid – it should be 12 digits not 11. A data-cleansing service running between the automation machine and the streaming encoder would have caught and rectified that, or the CDN themselves may be able to fix it. The ISRC code is used for royalty tracking and is typically not displayed in listener’s players.

RTP

Though it is technically possible to convey metadata over RTP, that capability is not in widespread practice, and any implementation of it would be a custom, in-house solution. Wheatstone Streaming Encoders do not support transport of metadata over RTP.

TRANSFORM FILTERS: THE MISSING ELEMENT

This brings us now to the question of how, exactly, is the incoming raw metadata from the automation system parsed and reformatted for transmission to the CDN? The answer is an imbedded, programmable scripting language called Lua (Portuguese for moon). Lua excels at the kind of search and pattern-matching tasks required to extract the desired information from the incoming text (which contains, as you might have noticed, a lot of extraneous "noise"), and then to rearrange it according to the dictates of its ultimate destination.

Within the Destination Pane is a button labeled METADATA PROCESSING. Open this to configure and enable metadata handling. Each destination stream has its own metadata settings and configuration. Each transport protocol also requires a different set of parameters for proper operation. But in all cases there is a Lua transform filter involved. Wheatstone streaming encoders come with a collection of transform filters that have been created for various combinations of three distinct players: (1) the automation system (affects what the incoming metadata consists of), (2) the stream transport type (affects how the metadata is transmitted), and (3) the destination CDN (affects which pieces of metadata are required and how they are to be presented).

So each transform filter is a little stand-alone computer program that carries out a specific transformation of input to output. All of the filters contain a function called `parseMetaData()`, which is where the action begins. This function accepts a string of raw metadata, from which it extracts particular elements of the metadata by means of a function called `findMatch()`. These elements are collected into local variables, which are then used to build an output string which is returned to the streaming encoder.

One common element that all of the filters must contend with is text encoding. UTF-8 has become the *de facto* standard for digital text, but percent-encoding (also called URL- or URI-encoding) of certain characters will continue to be mandatory for URLs (for example in the HTTP messages sent to Icecast servers), and HTML documents will continue to use "HTML Entities" to encode special characters. For this reason there are utility functions for text conversion included in many of the filters. For example, the function `htmlEntity_to_url()` converts HTML Entities (such as "&amp;") to their URL-encoded equivalents ("%26"), and the function `htmlEntity_to_utf8()` does the same for UTF-8 output. There are also similar functions for `utf8_to_url()` and `url_to_utf8()`. (We never have to encode HTML Entities, so there are no functions for doing that.)

There are other utility functions as well. The functions `getDurMSec()` and `getDurSec()` convert durations in *hh:mm:ss* format to milliseconds and seconds, respectively. There are also versions of each of these that do the same for URL-encoded durations: *hh%3Amm%3Ass*. Filters for HLS streams contain special functions for creating binary representations of ID3v2 frames. Many of the filters call the `trim()` function to remove whitespace from the beginnings and endings of elements such as title, artist and album. And there are actually two versions of `findMatch()` -- one for locating elements in XML data, and the other for identifying key/value pairs.

Though a complete description of the Lua language is well beyond the scope of this document [for that, the best place to start is at www.lua.org], we can point out a few things which might ease your introduction. Comments begin with a sequence of two dashes (hyphens) and last through the end of the line. Variables are instantiated and declared simply by naming them. Their type (integer, decimal, string) is fluid and determined by context. Logical expressions rely on keywords rather than punctuation symbols, for example "`if…then…end`" and "`if…and…or…then…end`". Familiar symbols are used for assignment (=), equality (==), and non-equality (~=). The concatenation operator (..) makes it easy to glue pieces of a string together.

One interesting peculiarity of the Lua language is that functions may return more than one value, separated by commas, to the caller. This feature is used by the `parseMetaData()` function to signal the type of transport mechanism the filter is designed for, as well as an optional metadata delay, in addition to the reformatted metadata itself. These details will be discussed further below.

All of the filters have access to "user variables," which are strings or numeric values set using the text fields on the left side of the window. Four of these user variables are specific to Icecast streams; these are SERVERURL, MOUNTPNT, USERNAME and PASSWORD. The other four variables are available to all filters, and are named USERVAL1 through USERVAL4. When a filter is loaded, the values in these fields are used to initialize the values of corresponding global variables in the filter. Whenever the value of a user variable is changed (either through manual text entry or by means of the scheduler), the corresponding global variable in the filter is also updated.

All of the filters may be edited and customized to your particular needs. The factory templates should be seen as working examples and useful starting points for further refinement.

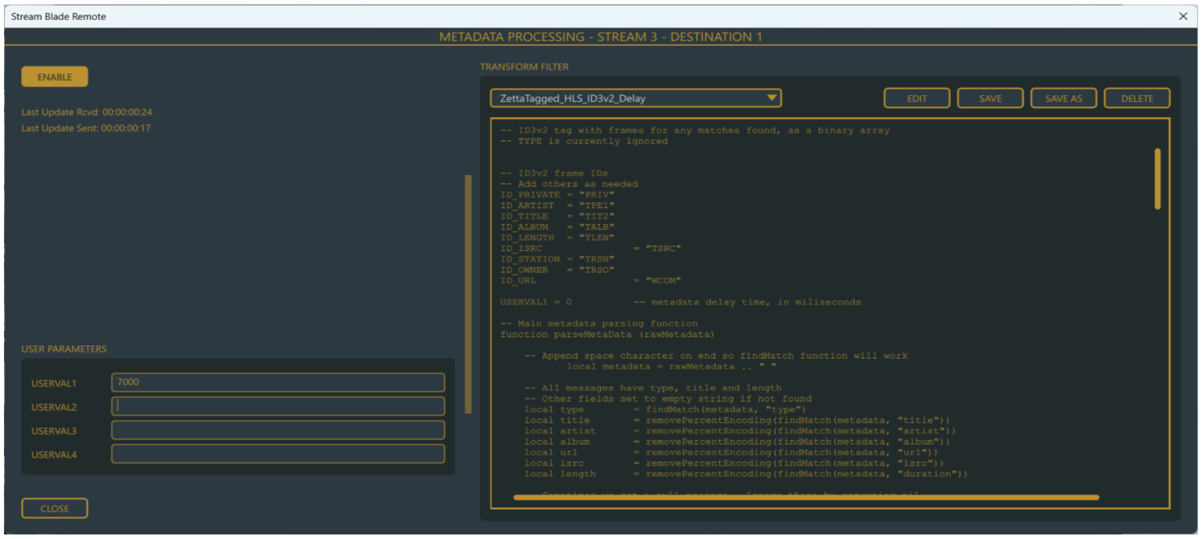
METADATA DELAY

Many radio stations, especially in the United States, employ a broadcast delay (a.k.a. profanity delay) during certain programs. If there is metadata being generated during these periods, then that metadata must also be delayed by the same amount if it is to remain in sync with the audio.

A common use of the user variables is to control metadata delay, by assigning a number to USERVAL1. (Any of the USERVALs may be used for this purpose.) This value is accessible by the filter and can then be added to the return value sequence as the very last element, where it will be interpreted by the Streaming Encoder as a delay in milliseconds. All filters, for any stream transport type, may use this mechanism to effect delays.

HLS

Filters designed for HLS streams have access to USERVALs 1-4. In this example, USERVAL1 is used to set a 7-second delay.



Last Update Rcvd: 00:00:00:24
Last Update Sent: 00:00:00:17

ZettaTagged_HLS_ID3v2_Delay

USERVAL1	7000
USERVAL2	
USERVAL3	
USERVAL4	

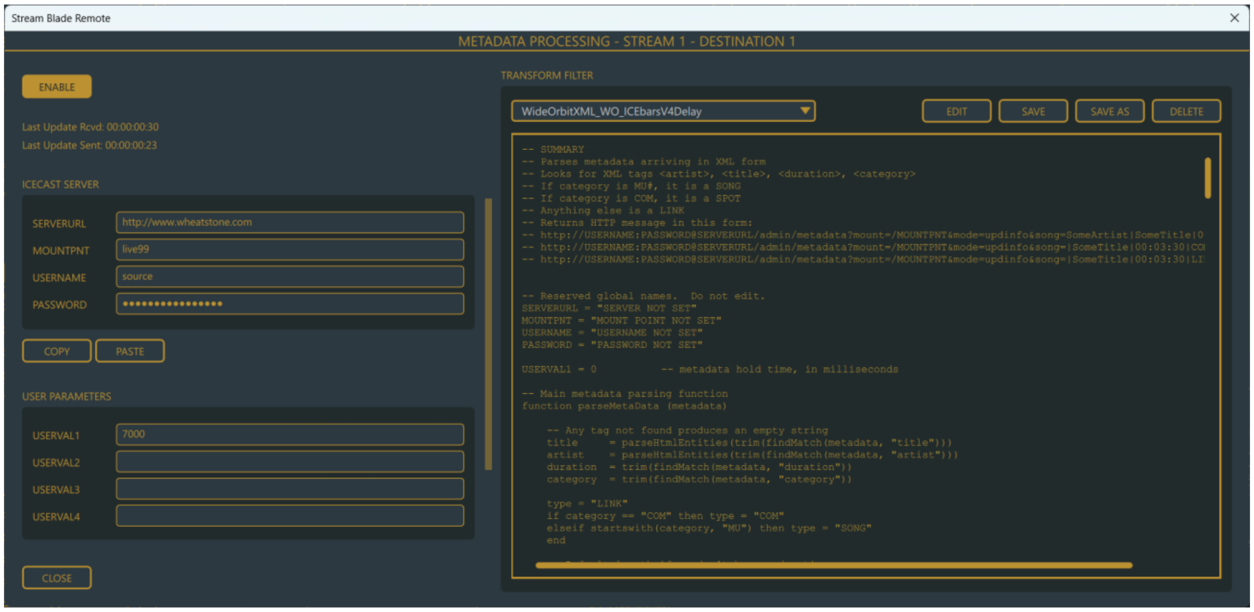
```
-- ID3v2 tag with frames for any matches found, as a binary array
-- TYPE is currently ignored


-- ID3v2 frame IDs
-- Add others as needed
ID_PRIVATE = "PRIV"
ID_ARTIST  = "TPE1"
ID_TITLE   = "TIT2"
ID_ALBUM   = "TALB"
ID_LENGTH  = "TLEN"
ID_ISRC               = "TSRC"
ID_STATION = "TRSN"
ID_OWNER   = "TRSO"
ID_URL                = "WCOM"

USERVAL1 = 0          -- metadata delay time, in miliseconds

-- Main metadata parsing function
function parseMetaData (rawMetadata)

    -- Append space character on end so findMatch function will work
           local metadata = rawMetadata .. " "

    -- All messages have type, title and length
    -- Other fields set to empty string if not found
    local type        = findMatch(metadata, "type")
    local title       = removePercentEncoding(findMatch(metadata, "title"))
    local artist      = removePercentEncoding(findMatch(metadata, "artist"))
    local album       = removePercentEncoding(findMatch(metadata, "album"))
    local url         = removePercentEncoding(findMatch(metadata, "url"))
    local isrc        = removePercentEncoding(findMatch(metadata, "isrc"))
    local length      = removePercentEncoding(findMatch(metadata, "duration"))
```

HLS transform filters have the primary task of transforming incoming metadata into an ID3v2 frame, which is a hybrid binary/text structure. HLS filters are required to return at least two values: the string "ID3" followed by the ID3v2 frame (which is itself a compound object composed of the ID3 tag header concatenated onto an array of individual ID3 frames). An optional third value specifies a delay in milliseconds. In this example, note the commas separating the multiple return values:

```
return "ID3" , mkTagHeader(frmSize, 4, 0) .. frmArray , USERVAL1
```

ICY

Filters designed for Icecast streams have access to 8 global variables: SERVERURL, MOUNTPNT, USERNAME and PASSWORD, in addition to USERVAL1-4. Icecast metadata is updated by means of HTTP messages sent independently of the audio stream. The target of the update message is usually (but not always) the same server which receives the audio stream, with the same authentication credentials. For this reason we have provided Copy/Paste buttons to transfer server settings from the Icecast Destination Pane into the metadata processing pane.



Last Update Rcvd: 00:00:00:30
Last Update Sent: 00:00:00:23

SERVERURL: http://www.wheatstone.com
MOUNTPNT: live99
USERNAME: source
PASSWORD: ••••••••••••••••

USERVAL1: 7000
USERVAL2:
USERVAL3:
USERVAL4:

WideOrbitXML_WO_ICEbarsV4Delay

```
-- SUMMARY
-- Parses metadata arriving in XML form
-- Looks for XML tags <artist>, <title>, <duration>, <category>
-- If category is MU#, it is a SONG
-- If category is COM, it is a SPOT
-- Anything else is a LINK
-- Returns HTTP message in this form:
-- http://USERNAME:PASSWORD@SERVERURL/admin/metadata?mount=/MOUNTPNT&mode=updinfo&song=SomeArtist|SomeTitle|0
-- http://USERNAME:PASSWORD@SERVERURL/admin/metadata?mount=/MOUNTPNT&mode=updinfo&song=|SomeTitle|00:03:30|CO
-- http://USERNAME:PASSWORD@SERVERURL/admin/metadata?mount=/MOUNTPNT&mode=updinfo&song=|SomeTitle|00:03:30|LI


-- Reserved global names.  Do not edit.
SERVERURL = "SERVER NOT SET"
MOUNTPNT = "MOUNT POINT NOT SET"
USERNAME = "USERNAME NOT SET"
PASSWORD = "PASSWORD NOT SET"

USERVAL1 = 0          -- metadata hold time, in milliseconds

-- Main metadata parsing function
function parseMetaData (metadata)

    -- Any tag not found produces an empty string
    title     = parseHtmlEntities(trim(findMatch(metadata, "title")))
    artist    = parseHtmlEntities(trim(findMatch(metadata, "artist")))
    duration  = trim(findMatch(metadata, "duration"))
    category  = trim(findMatch(metadata, "category"))

    type = "LINK"
    if category == "COM" then type = "COM"
    elseif startswith(category, "MU") then type = "SONG"
    end
```

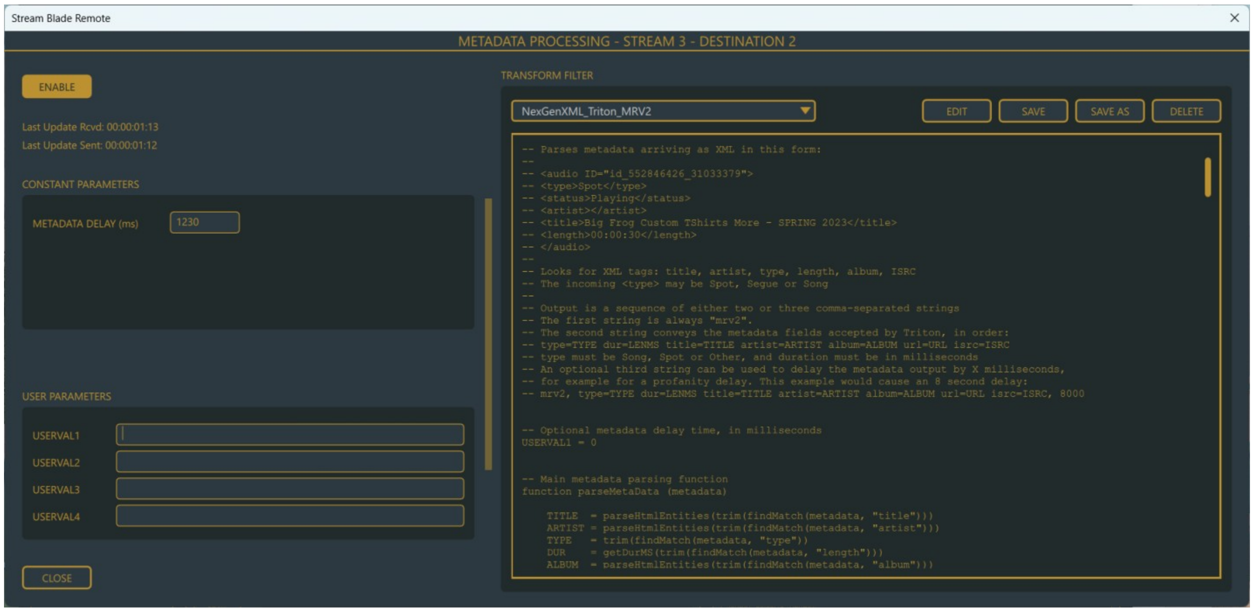
Icecast transform filters return one or more string values beginning with `http://` or `https://`, optionally followed by a last value which is interpreted as metadata delay, in milliseconds.

Cirrus Securenet Customers: Securenet Systems has implemented an augmented Icecast protocol, in which basic artist/title updates are sent to the audio server, while a fuller version of the metadata is sent to a second "Data Collection Server," with its own URL and authentication credentials. Thus a Securenet transform filter returns two HTTP strings for each incoming metadata event. See Chapter 7 for details.

TuneIn Radio Customers: Once your URL is entered into the SERVERURL field, the labels on the remaining required fields change to STATION_ID, PARTNER_ID and PARTNER_KEY.

MRV2

Filters designed for Triton streams have access to the four USERVALs, plus a special METADATA DELAY setting which is used internally for precise audio/metadata synchronization. This synchronization delay is separate from a broadcast delay, which, like the other transport types, may be signaled by adding it as a last return value. See Chapter 6 for further details.



Last Update Rcvd: 00:00:01:13
Last Update Sent: 00:00:01:12

CONSTANT PARAMETERS

METADATA DELAY (ms): 1230

USER PARAMETERS

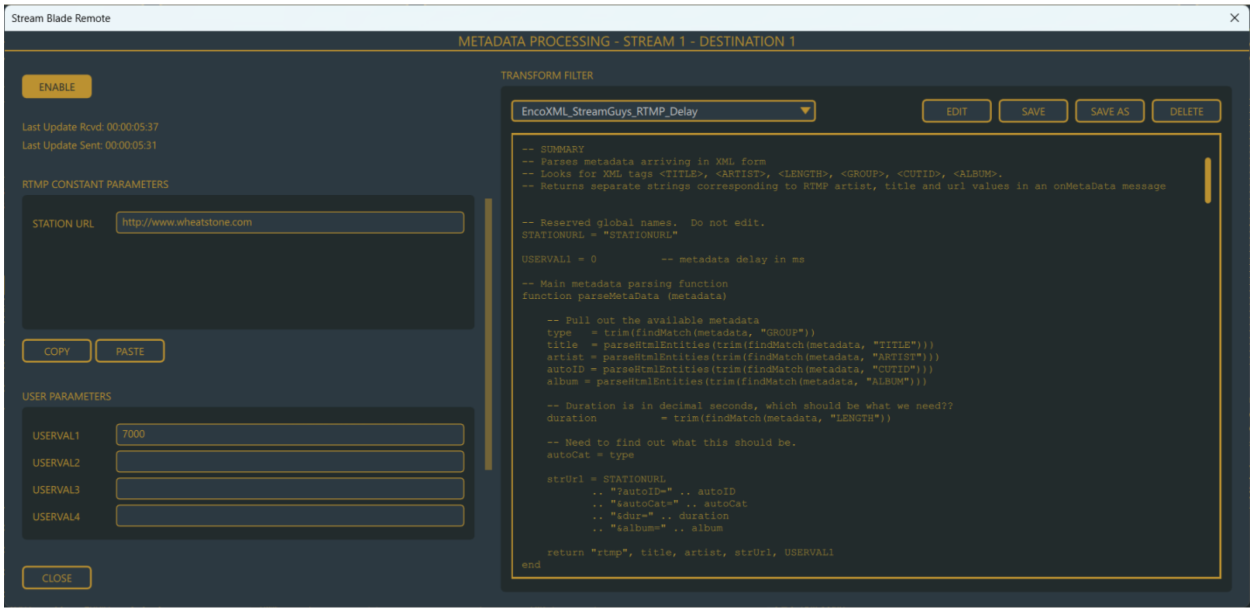
USERVAL1:
USERVAL2:
USERVAL3:
USERVAL4:

NexGenXML_Triton_MRV2

```
-- Parses metadata arriving as XML in this form:
--
-- <audio ID="id_552846426_31033379">
-- <type>Spot</type>
-- <status>Playing</status>
-- <artist></artist>
-- <title>Big Frog Custom TShirts More - SPRING 2023</title>
-- <length>00:00:30</length>
-- </audio>
--
-- Looks for XML tags: title, artist, type, length, album, ISRC
-- The incoming <type> may be Spot, Segue or Song
--
-- Output is a sequence of either two or three comma-separated strings
-- The first string is always "mrv2".
-- The second string conveys the metadata fields accepted by Triton, in order:
-- type=TYPE dur=LENMS title=TITLE artist=ARTIST album=ALBUM url=URL isrc=ISRC
-- type must be Song, Spot or Other, and duration must be in milliseconds
-- An optional third string can be used to delay the metadata output by X milliseconds,
-- for example for a profanity delay. This example would cause an 8 second delay:
-- mrv2, type=TYPE dur=LENMS title=TITLE artist=ARTIST album=ALBUM url=URL isrc=ISRC, 8000


-- Optional metadata delay time, in milliseconds
USERVAL1 = 0


-- Main metadata parsing function
function parseMetaData (metadata)

    TITLE  = parseHtmlEntities(trim(findMatch(metadata, "title")))
    ARTIST = parseHtmlEntities(trim(findMatch(metadata, "artist")))
    TYPE   = trim(findMatch(metadata, "type"))
    DUR    = getDurMS(trim(findMatch(metadata, "length")))
    ALBUM  = parseHtmlEntities(trim(findMatch(metadata, "album")))
```

RTMP

Filters designed for RTMP streams have access the four USERVALs, as well as a STATIONURL field which is used in the formulation of the metadata update injected into the stream. The STATIONURL field should be a real URL, preferably one which points to your station's home page, in the form `http://www.mystation.com`.



Last Update Rcvd: 00:00:05:37
Last Update Sent: 00:00:05:31

RTMP CONSTANT PARAMETERS

STATION URL: http://www.wheatstone.com

USER PARAMETERS

USERVAL1: 7000
USERVAL2:
USERVAL3:
USERVAL4:

EncoXML_StreamGuys_RTMP_Delay

```
-- SUMMARY
-- Parses metadata arriving in XML form
-- Looks for XML tags <TITLE>, <ARTIST>, <LENGTH>, <GROUP>, <CUTID>, <ALBUM>.
-- Returns separate strings corresponding to RTMP artist, title and url values in an onMetaData message


-- Reserved global names.  Do not edit.
STATIONURL = "STATIONURL"

USERVAL1 = 0          -- metadata delay in ms

-- Main metadata parsing function
function parseMetaData (metadata)

    -- Pull out the available metadata
    type   = trim(findMatch(metadata, "GROUP"))
    title  = parseHtmlEntities(trim(findMatch(metadata, "TITLE")))
    artist = parseHtmlEntities(trim(findMatch(metadata, "ARTIST")))
    autoID = parseHtmlEntities(trim(findMatch(metadata, "CUTID")))
    album = parseHtmlEntities(trim(findMatch(metadata, "ALBUM")))

    -- Duration is in decimal seconds, which should be what we need??
    duration          = trim(findMatch(metadata, "LENGTH"))

    -- Need to find out what this should be.
    autoCat = type

    strUrl = STATIONURL
           .. "?autoID=" .. autoID
           .. "&autoCat=" .. autoCat
           .. "&dur=" .. duration
           .. "&album=" .. album

    return "rtmp", title, artist, strUrl, USERVAL1
end
```

CUSTOMIZING METADATA TRANSFORM FILTERS

Click the Transform Filter EDIT button to enable text changes. If you edit a factory filter then saving it will create a new filter. Any time a filter is saved it is also reloaded, so check the top of the Status Pane in the main window to confirm you have no syntax errors.

When editing a filter it is often useful to transmit metadata to it immediately, to confirm that your changes work as expected. We have found a free utility called Packet Sender (https://packetsender.com) to be extremely useful in this regard. It provides a simple interface to paste in ASCII text and send it as either a TCP or UDP message to any address/port.

Filter which you create will appear at the top of the transform filter drop-down list in white text. Factory templates are listed below in gold text.

CHAPTER 4 - USING THE SCHEDULER

Streaming your programming is easy, right? Sure, until it's not.

While most stations will stream whatever is on their terrestrial on-air signal all of the time, performance rights rules sometimes makes things complicated. For example, if your station carries sporting events, some leagues won't allow you to stream that programming. Or perhaps your station streams local sporting events that aren't carried on your on air signal several times a week, or your station carries a mix of music and talk and you want to optimize the stream's processing to match the source material.

With the scheduling software that's built into Wheatstream, you don't need to count on a board operator going in to click the right button at the right time.

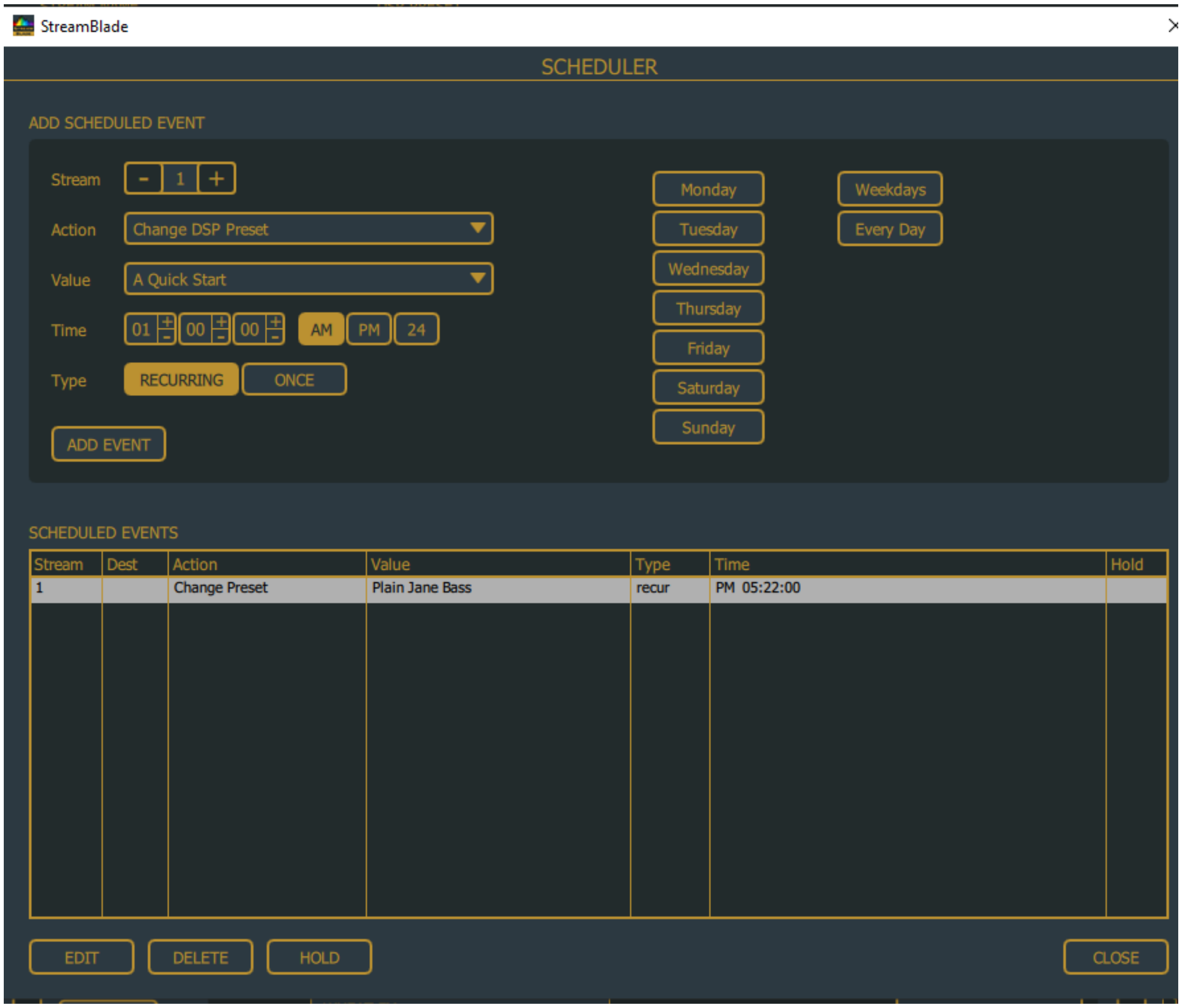
EVENTS THAT YOU CAN SCHEDULE

- Change DSP Preset
- Start Destination Stream
- Stop Destination Stream
- Set Metadata Filter (4 User Variables available)

Events may be scheduled to happen once at a specific day and time, or to happen on a recurring basis. Recurring events can happen at a time on a specific day of the week, or at a specific time on weekdays only, or at the same time every day.

CREATE A RECURRING EVENT

Click the Scheduler button in the lower left corner.



Stream: 1

Action: Change DSP Preset

Value: A Quick Start

Time: 01 00 00 [✓ AM] PM 24

Type: [✓ RECURRING] ONCE

Stream	Dest	Action	Value	Type	Time	Hold
1		Change Preset	Plain Jane Bass	recur	PM 05:22:00	

Choose which stream number you are using.

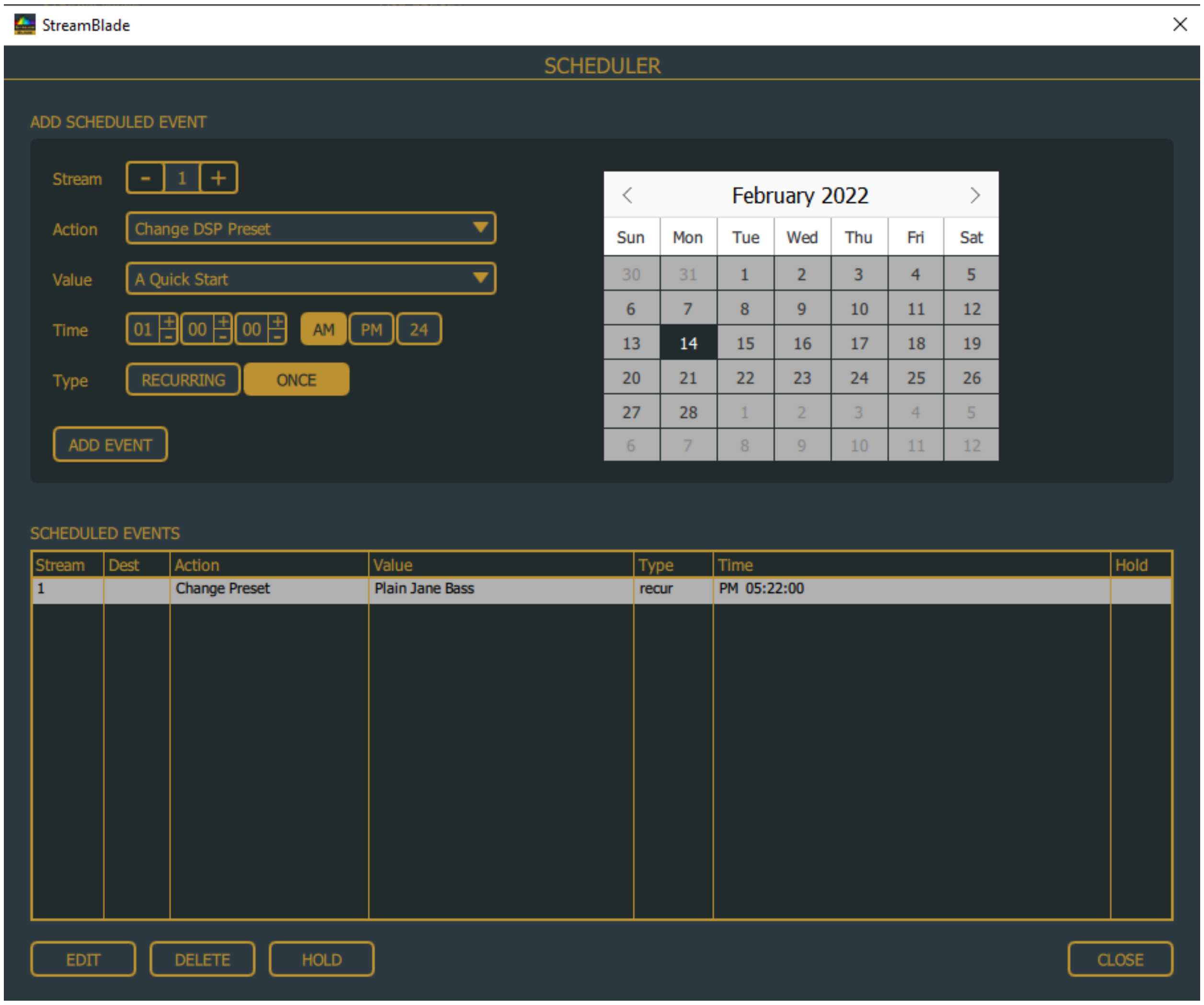
Pick the **Action** from the drop-down menu. The **value** field will change to match the action.

Set the time, using the buttons to select AM/PM or 24 hour time entry.

Click the buttons for the days of the week it will repeat. You can choose more than one day of the week. The Weekdays and Every Day buttons may be used as a shortcut. Click **Add Event** to add this to the schedule.

CREATE A ONE-TIME EVENT

Click the Scheduler button in the lower left corner.



Stream: 1

Action: Change DSP Preset

Value: A Quick Start

Time: 01 00 00 [✓ AM] PM 24

Type: RECURRING [✓ ONCE]

February 2022

Sun	Mon	Tue	Wed	Thu	Fri	Sat
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	[✓ 14]	15	16	17	18	19
20	21	22	23	24	25	26
27	28	1	2	3	4	5
6	7	8	9	10	11	12

Stream	Dest	Action	Value	Type	Time	Hold
1		Change Preset	Plain Jane Bass	recur	PM 05:22:00	

Choose which stream number you are using.

Pick the **Action** from the drop-down menu. The **value** field will change to match the action.

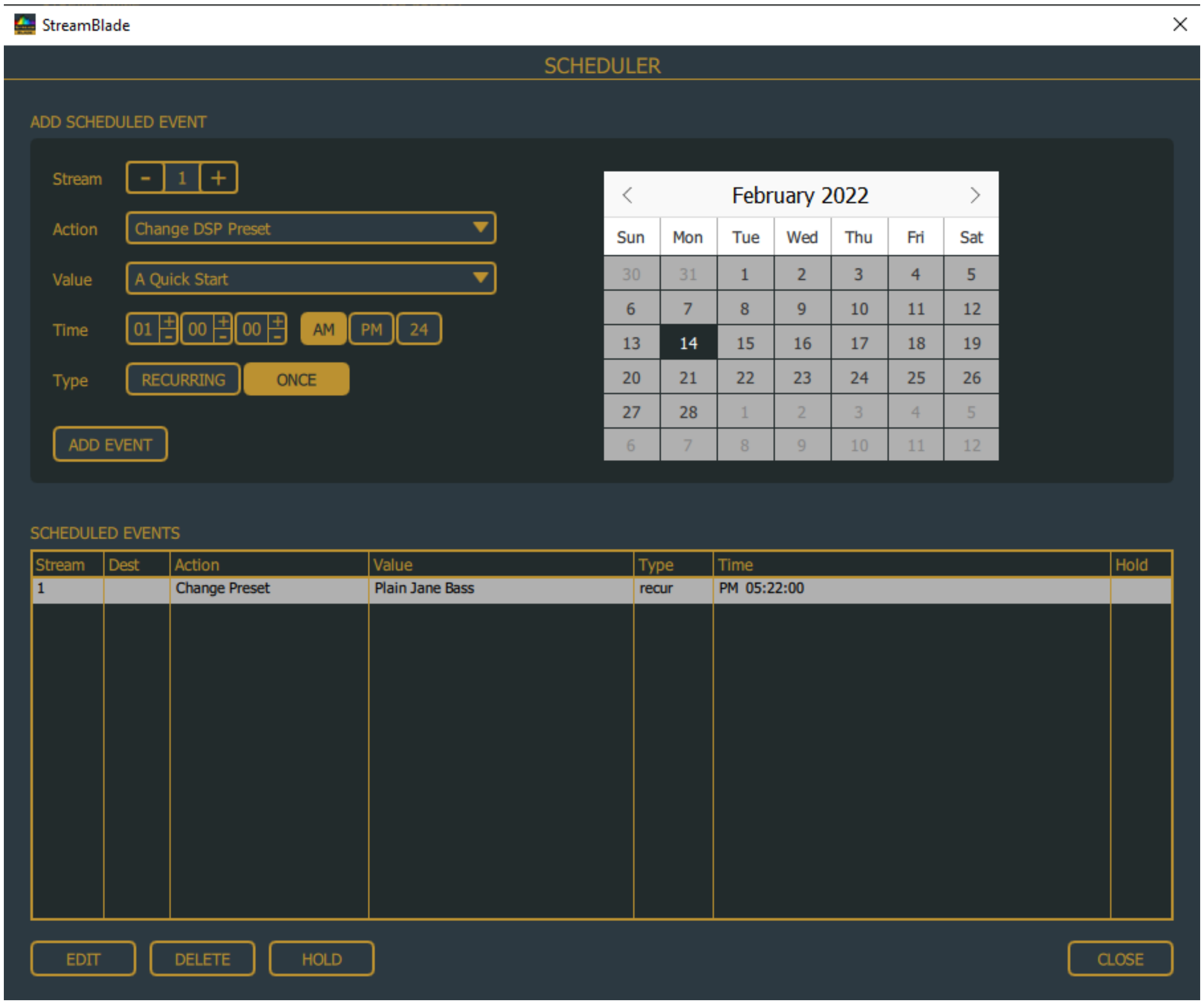
Set the time, using the buttons to select AM/PM or 24 hour time entry.

Click the **ONCE** button. A calendar will appear. Highlight the desired date.

Click **ADD EVENT** to add this to the schedule.

PUT A SCHEDULED EVENT ON HOLD

Click the Scheduler button in the lower left corner



Stream: 1
Action: Change DSP Preset
Value: A Quick Start
Time: 01 00 00 [✓ AM] PM 24
Type: RECURRING [✓ ONCE]

Stream	Dest	Action	Value	Type	Time	Hold
1		Change Preset	Plain Jane Bass	recur	PM 05:22:00	

Click to highlight the desired event.

Click the **HOLD** button.

The word “true” will appear in the Hold column.

Events on hold will be skipped. To resume the schedule for this event, click to highlight the event, click the **HOLD** button, and the “true” will be removed making the event active again.

USING THE METADATA FILTER TO DELAY METADATA

When your station uses profanity delay, you can get into a situation where the title and artist metadata will be sent to the stream encoder before the audio actually hits.

If your automation system has a Lua metadata transform filter for delay, setting the user value of that filter to match the profanity delay buffer time will delay the metadata by that many seconds.

For example, a Wide Orbit customer whose User Defined Filter of *WideOrbitXML_WideOrbitICEbarsV3Delay* using a 12 second profanity delay set the value of that filter to 12, the metadata change would be delayed by 12 seconds.

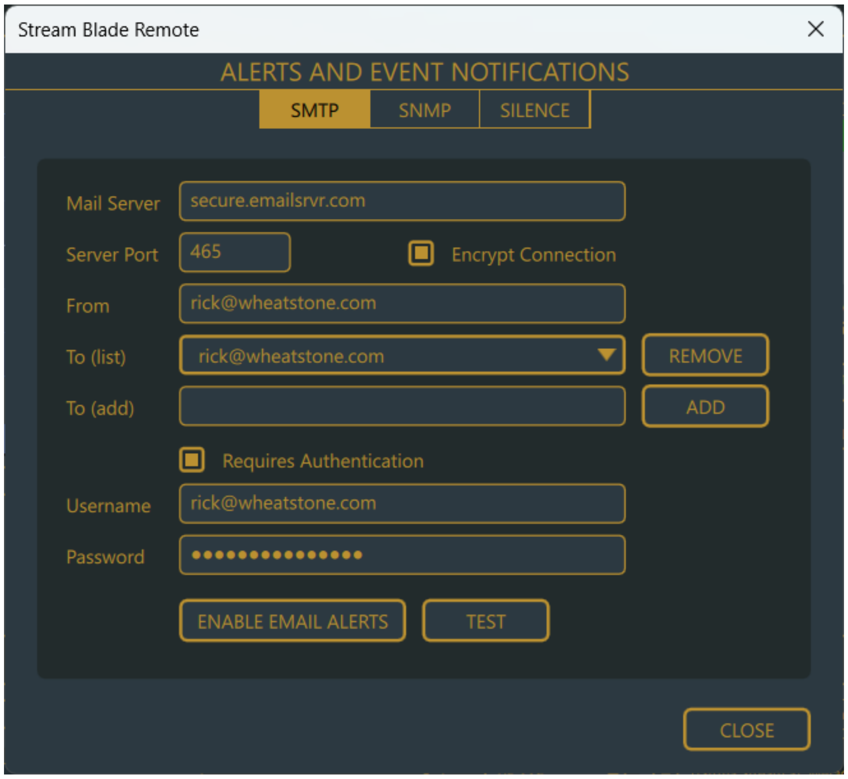
If you're only in delay for one daypart like a morning show, putting this filter on a schedule would delay the title/artist information while you're in delay and then shift back to normal when the profanity delay is removed.

See Chapter 4 to learn how to upload Lua metadata transform filters.

CHAPTER 5 – ALERTS AND STATUS MONITORING

EMAIL (SMTP)

Email alerts may be set up to send notifications about socket errors, send and receive errors, silence alerts (and cancellations when signal is restored), stream starting and starting, stream disconnects and restart attempts. Open relay servers are supported, as well as most secure servers. Microsoft Office 365 accounts are not currently supported.



Mail Server: secure.emailsrvr.com

Server Port: 465

[✓] Encrypt Connection

From: rick@wheatstone.com

To (list): rick@wheatstone.com

To (add):

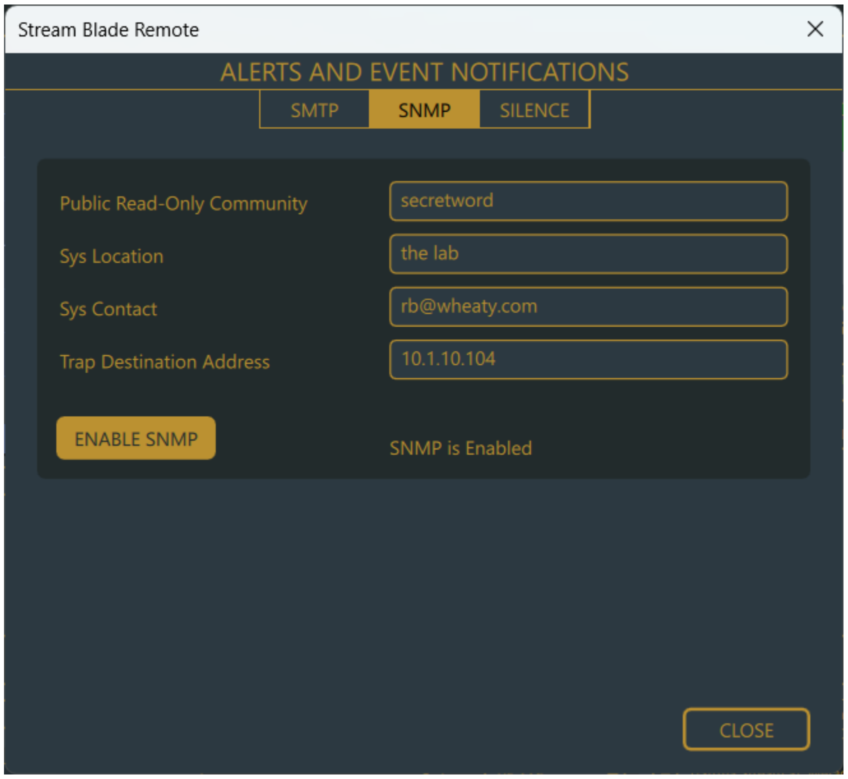
[✓] Requires Authentication

Username: rick@wheatstone.com

Password: •••••••••••••••

SNMP

SNMP can be configured to monitor Wheatstream status as well as to receive event notifications, or “traps,” as they are referred to in the argot of SNMP. Four items of information are required to configure SNMP: (1) a “read-only community string,” which is nothing more than a password that you create yourself; (2) the physical location of the Wheatstream unit; (3) email address of the system administrator; and (4) the IP address of the machine designated to receive traps (use the address 0.0.0.0 to disable traps).

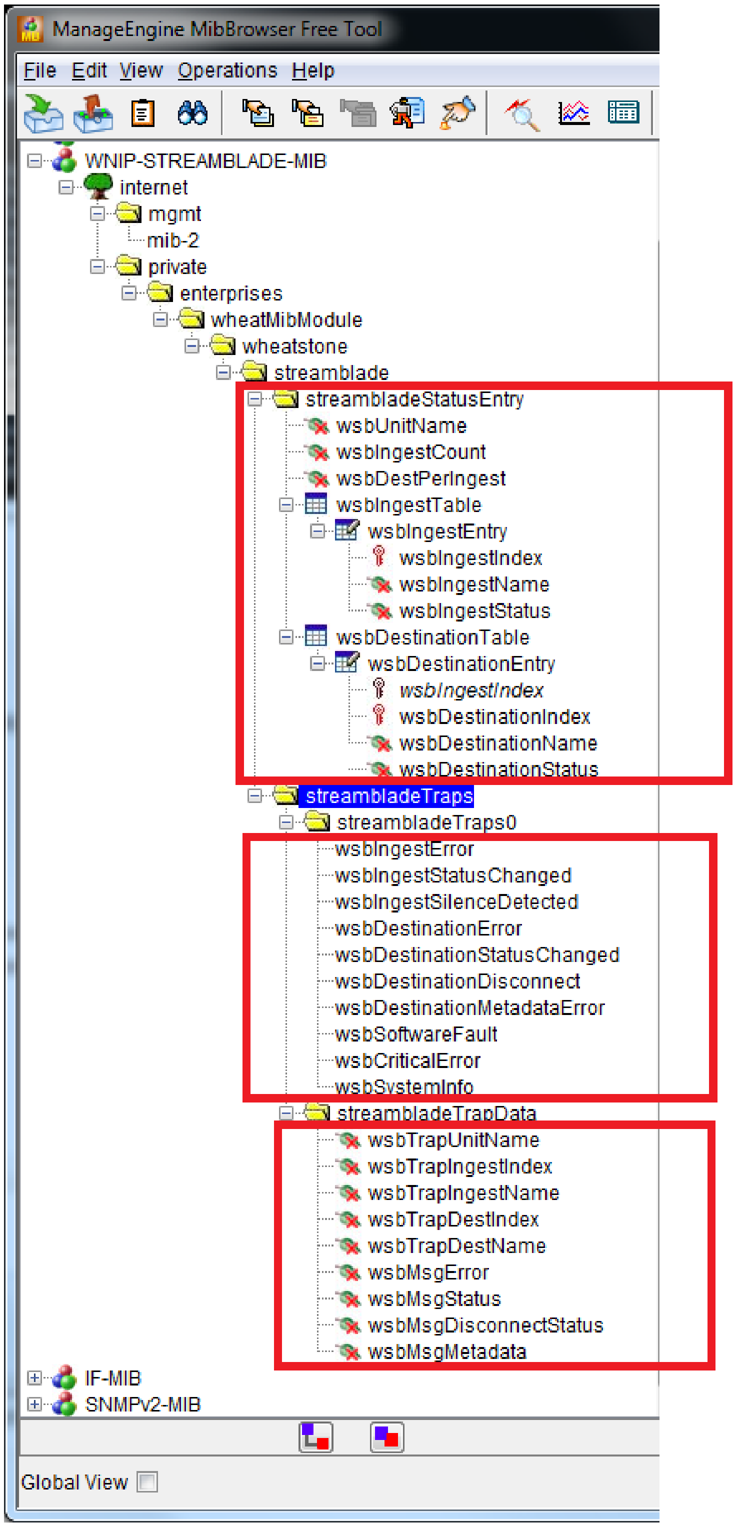

Stream Blade Remote

ALERTS AND EVENT NOTIFICATIONS

Public Read-Only Community	secretword
Sys Location	the lab
Sys Contact	rb@wheaty.com
Trap Destination Address	10.1.10.104

SNMP is Enabled

The structure of the data available from a particular application via SNMP is specified in a MIB file (Management Information Base), which is a standard text file.   Wheatstream's MIB is called WNIP-STREAMBLADE-MIB, and may be loaded into any MIB browser. In the SNMP model, Wheatstream would be called the "agent", and your MIB browser is the "client." In the annotated screenshot below, we are using the ManageEngine MibBrowser to display the hierarchical structure of available data, status types, and traps.


ManageEngine MibBrowser Free Tool

File Edit View Operations Help

```
WNIP-STREAMBLADE-MIB
  internet
    mgmt
      mib-2
    private
      enterprises
        wheatMibModule
          wheatstone
            streamblade
              streambladeStatusEntry
                wsbUnitName
                wsbIngestCount
                wsbDestPerIngest
                wsbIngestTable
                  wsbIngestEntry
                    wsbIngestIndex
                    wsbIngestName
                    wsbIngestStatus
                wsbDestinationTable
                  wsbDestinationEntry
                    wsbIngestIndex
                    wsbDestinationIndex
                    wsbDestinationName
                    wsbDestinationStatus
              streambladeTraps
                streambladeTraps0
                  wsbIngestError
                  wsbIngestStatusChanged
                  wsbIngestSilenceDetected
                  wsbDestinationError
                  wsbDestinationStatusChanged
                  wsbDestinationDisconnect
                  wsbDestinationMetadataError
                  wsbSoftwareFault
                  wsbCriticalError
                  wsbSystemInfo
                streambladeTrapData
                  wsbTrapUnitName
                  wsbTrapIngestIndex
                  wsbTrapIngestName
                  wsbTrapDestIndex
                  wsbTrapDestName
                  wsbMsgError
                  wsbMsgStatus
                  wsbMsgDisconnectStatus
                  wsbMsgMetadata
IF-MIB
SNMPv2-MIB
```

Global View

This is status info that must be polled.

Table of ingest streams, with status (running/stopped/error/etc)

Table of destinations (CDN streams), per ingest, with status of each.

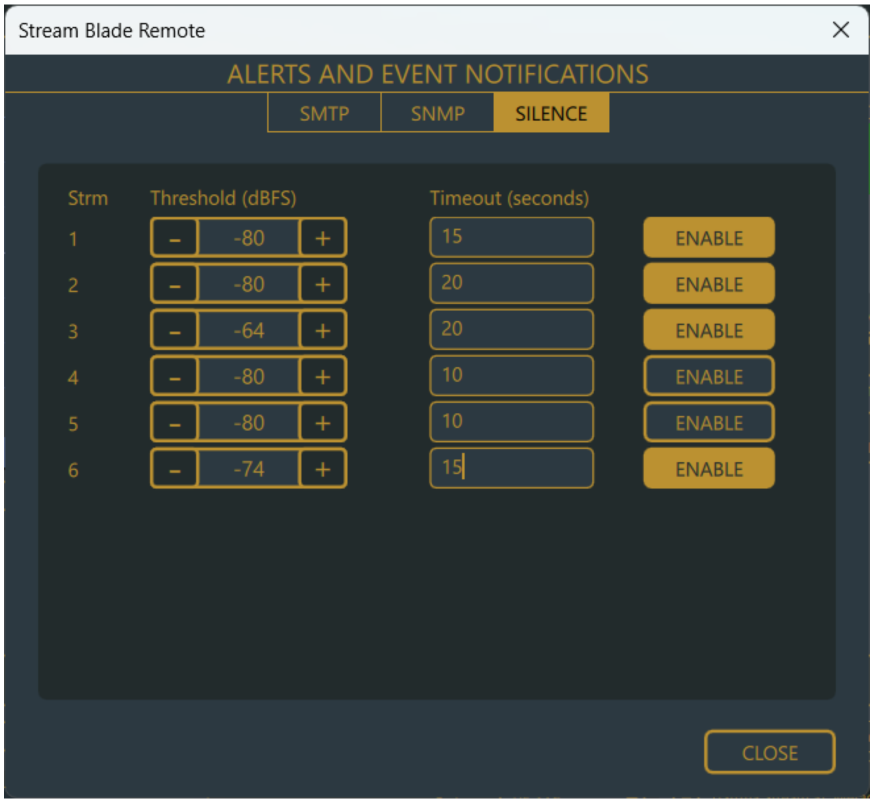
These are error and status change notification types.

This is the information contained therein.

Note that, for security purposes, Wheatstream does not provide access to a “read-write community string.” In other words, you may not use SNMP to control or change any settings in Wheatstream. Information flows in one direction only.

SILENCE

The silence detection feature monitors input levels on a per-ingest-stream basis. There are only two variables to consider: (1) the threshold that defines "silence," and (2) how long the signal needs to remain below that threshold in order for an alert to be triggered. Silence alerts are enabled by default, but may be disabled entirely for special situations. Note that "silence" is not the same thing as "no input." A silent ingest stream is still an active stream – that is, data is flowing in, it just happens to consist of all zeros (or very small numbers). An ingest stream which *stops* is considered to be a different type of error.



Strm	Threshold (dBFS)	Timeout (seconds)	
1	-80	15	ENABLE
2	-80	20	ENABLE
3	-64	20	ENABLE
4	-80	10	ENABLE
5	-80	10	ENABLE
6	-74	15	ENABLE

CHAPTER 6 – STREAMING TO TRITON DIGITAL

Sending a stream to Triton Digital is accomplished through Triton's proprietary MRV2 protocol, which is fully supported by all Wheatstone streaming applications. The MRV2 protocol imposes certain restrictions in how it may be used; the tradeoff being a tighter synchronization between audio and metadata, as well as automatic provisioning of streams which takes the guesswork out of setup and configuration. In the following discussion it is assumed that you already have an account with Triton Digital, and that you have received login information from Triton for one or more stations along with their associated passwords.

To begin, select an ingest stream that you wish to send to Triton. In the Destinations pane, select the MRV2 protocol. Note that this disables the URL, Mount, User, Password and Port fields, as well as the Codec and Bitrate dropdown lists. Click the Provisioning button which opens the Triton Provisioning dialog. Enter the station name provided by Triton into the ADD NEW field, and then click the ADD button. This adds the station to the STATION dropdown list. Enter the station password into the PASSWORD field and click the GET CURRENT PROVISIONING DATA button. If there is an error with either station name or password you will see a message to that effect in the Status log. Assuming your credentials are valid, the AVAILABLE MOUNTS list in the lower half of the dialog will populate. The number of mounts that you receive has already been determined by Triton in consultation with you when you set up your account.

Each mount corresponds to a specific codec and bitrate. Note that a sample-rate is specified as well, so confirm that this rate matches the sample-rate of your AoIP network (or more precisely, of your ingest stream). There is also a column called Backups which we'll get to in a moment.

At this point you need to decide how you want to apply your available mounts to specific Streams and Destinations. Note that the Wheatstream architecture provides a maximum of four destination streams (CDN streams) for each ingest, so as long as you have four mounts or less, then they can all be run off of the same ingest stream. But if you have five to eight mounts, you will need to spread them across at least two ingest streams, and in the rare case of more than eight mounts you will need to use at least three ingest streams to feed all of them.

So, select the mounts you wish to apply by clicking the check boxes in the right-most column of the table, then click the APPLY MOUNTS TO DESTINATIONS button. There are two important things to note here. First: mounts are applied in order to consecutive destinations beginning with the destination specified in the "Starting at Destination" control under the APPLY MOUNTS button (normally starting with Destination 1). Second: the destinations targeted by this operation are those which belong to the currently selected ingest stream, which is displayed at the top of the provisioning dialog.

To give a concrete example, let's say you have six mounts, which you want to divide between ingest Streams 2 and 3, using Destinations 1, 2 and 3 from Stream 2, and Destinations 2, 3 and 4 from Stream 3. So you would select Stream 2, then select three of your mounts and apply them starting at Destination 1. Next, you select Stream 3, then select your remaining mounts and apply them starting at Destination 2. You're done. Close the provisioning dialog and you will see that all six of your destination streams have been configured and should, in fact, be running at this point (assuming that audio is already being ingested into Streams 2 and 3).

Now that we've covered the basics, it's time to discuss a few of the finer points: metadata synchronization, redundancy, loudness control and continuous provisioning. Let's address these in order.

METADATA SYNCHRONIZATION

The entire point of the MRV2 protocol is to keep your audio and metadata – especially ad insertion triggers – tightly synchronized. This is accomplished in two ways: the metadata is actually injected into the audio stream, meaning that it cannot drift out of sync once injected, and the precise timing of the metadata is calibrated to the millisecond to create glitch-free ad insertions and returns to program content. In the Destinations pane, under the PROVISIONING button, there is another button labeled METADATA PROCESSING. Click this to open the Metadata Processing dialog. Select an MRV2 transform filter appropriate to your automation system.

Triton Digital has developed a method to determine the offset between the arrival of the metadata signaling an ad-insertion point and the actual break in the audio stream where this occurs. The procedure is outlined in this article:

https://tritondigital.my.site.com/s/article/Cue-Point-Alignment-with-Third-party-Appliances?language=en_US

Once the offset has been determined, enter the required delay in the METADATA DELAY field in the upper left of the Metadata Processing dialog.

REDUNDANT/HIGH AVAILABILITY STREAMS

Redundancy is built into the MRV2 protocol. In High Availability configuration, *multiple* streams are available simultaneously in order to provide uninterrupted service to listeners in the event of a break in connection anywhere between and including the station and the Triton streaming infrastructure.
It works by running multiple streams simultaneously, serving multiple listeners, where any given listener is only connected to one stream at a time. But if there is a break in the connection (from the station or within the Triton network), the listener is immediately and seamlessly connected to one of the other streams.

You can create a redundant/failsafe streaming setup by creating two identical hardware configurations on two separate Wheatstreams, each with its own input feed, and streaming the output of both to the Triton Digital Network. The Triton Media Relay will accept one of the streams and ignore the other. If one of your machines fails, or if one of the feeds is broken, the Media Relay will switch over to the other stream. For best redundant security, the two systems should be located on separate power lines, network cables, and ISPs.

If you can't install two machines, you can still benefit from some level of redundancy by assigning your station to two identical streams. To make this work: you need to create each stream **two times**, perfectly identical, with the same mounts, in the Streamblade Remote GUI.

You must inform Triton Digital if you decide to set up a redundant system so they can make the appropriate configuration on their side.

LOUDNESS CONTROL

You might have noticed the “Recommended Loudness Setting” displayed above the mounts table in the provisioning dialog. This is not enforced by the Wheatstream software, but it is recommended especially if you are doing ad insertions, because it keeps your program content at the same level as the ad content which will be inserted into the stream going out to listeners. Automatic loudness control is enabled in the DUALBAND LIMITER pane in the main window of the Wheatstream GUI.

CONTINUOUS PROVISIONING

Occasionally Triton needs to change certain parameters of a mount for load-balancing purposes. Fortunately this process is also handled for you automatically. Every few minutes, Wheatstream contacts the Triton production server to request current provisioning data. If there has been no change, then fine, nothing is done. However, if any parameters have changed for any of your running streams, they will automatically be stopped and restarted using the new settings. In rare cases a new mount might be created, or an existing mount removed. Any destination stream going to a mount which is removed will be stopped. If a new mount is created then you will need to manually apply it to a new destination. In any case, all such provisioning changes are recorded in the Status Log as they occur.

ADDITIONAL RESOURCES

Triton has kindly provided additional documentation which will be useful to Wheatstream customers who are planning to employ Triton’s streaming services.

Triton/Wheatstone Integration:

https://tritondigital.my.site.com/s/article/Using-Wheatstone-Encoders-with-the-Triton-Digital-Streaming-Network?language=en_US

Encoder Bitrates:

https://tritondigital.my.site.com/s/article/Choosing-Audio-Bitrate-Settings?language=en_US

CHAPTER 7 – STREAMING TO CIRRUS SECURENET

Wheatstone is developing a method to automatically provision metadata configuration for stream connections to Securenet Systems. Until that is available, the following manual steps must be taken.

Log into the Securenet "dashboard" at https://c360.cir.st

Your initial goal is to formulate a URL which will allow you to download an XML document which contains pertinent provisioning information.

Click the API & UTILITIES link, upper right.

Scroll to the bottom. You'll see a six-character one-time code under the App Login heading. Grab that and paste it at the end of this URL:

https://dx.cir.st/radio/config_get_info.cfm?trcode=<your-one-time-code>

A more permanent form of the provisioning request requires that you grab your Call Letters and Authorization Token which are displayed in the dashboard, and use this form:

https://dx.cir.st/radio/config_get_info.cfm?stationCallSign=<your-call-sign>&authtoken=<your-authorization-token>

Paste either of those URL forms into the address bar of your browser. This will return provisioning XML to you.

Towards the top of the XML you'll find the <encoder> tag. This in turn contains <stationCallSign> which is the Icecast "mount," and <serverAddress> which is the URL that your audio is going to. The user is "source." The password is the same one that you used to log into the dashboard (which is also repeated in the <password> tag). You also see the recommended bit rate, sample rate and encoder (AAC or MP3).

In the Metadata Processing dialog in Streamblade, there are two forms you need to fill in. The one at the top (ICECAST SERVER) contains the same information that you entered into the Destinations pane for this stream. (You can Copy it from the Destinations pane and paste into the Metadata Server form.) Any URL containing the string “securenetsystems” which is entered into the top form has the side effect of changing the labels in the USER PARAMETERS form, to DCSSERVER, CALLSIGN and AUTHTOKEN.

You’ll find the DCS (Data Capture Software) server in the provisioning XML a bit further down under the <DCS> tag, which in turn contains a <playerServer> tag with something like "https://streamdb7web.securenetsystems.net". Paste or type that into the DCSSERVER field. The CALLSIGN is the same as the MOUNTPNT above, and you also need to supply your authorization token in the AUTHTOKEN field.

The reason that we need both an Icecast URL as well as a DCS URL is that two different messages are sent to two different servers at Securenet Systems for each incoming metadata message. One is for normal “now playing” data displayed to the listener, while the other is for internal tracking of music royalties, ad insertions, logging, etc.

There is currently a small collection of metadata transform filters which demonstrate how metadata updates to Securenet are formatted and sent.

CHAPTER 8 – MAINTENANCE AND OPERATIONAL TIPS

MONITOR VIEW KEYBOARD SHORTCUTS

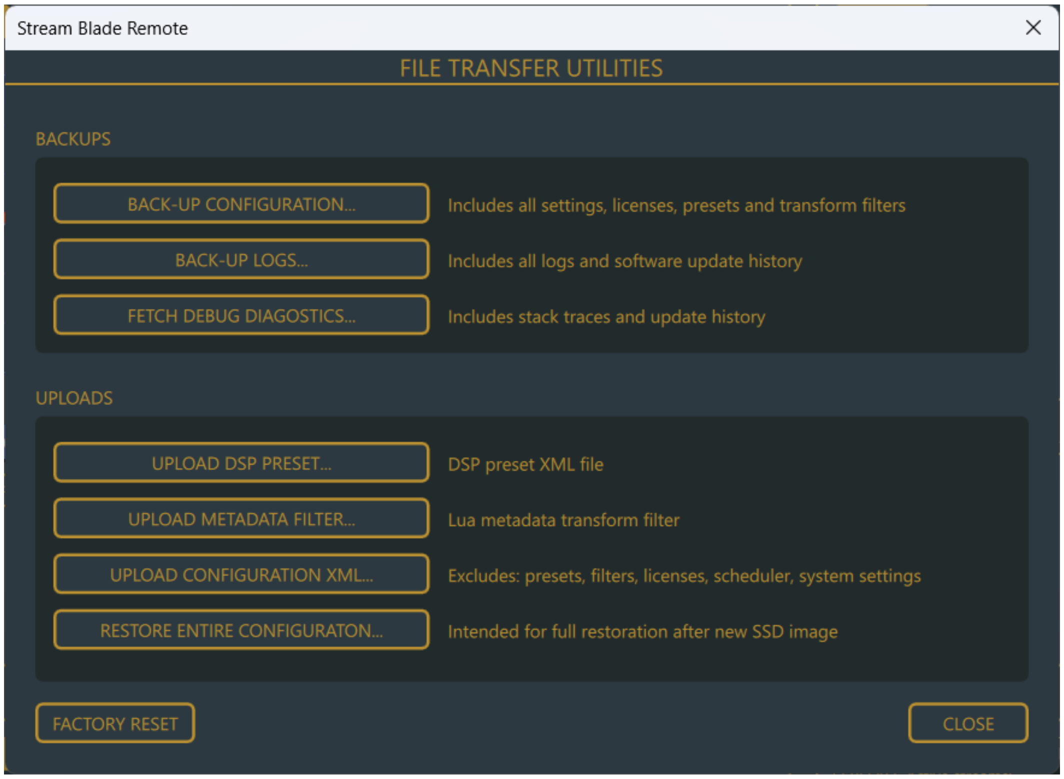
If you are using a less-than full size monitor for either server or remote client, the following keyboard shortcuts will be helpful:

<Alt> + F2 toggles in and out of fullscreen mode

<Alt> + F3 reduces the window to 80% of monitor size

REMOTE UTILITIES

Clicking the UTILITIES button in the lower left corner while in the remote client will give you the following options. Note that files downloaded from the Wheatstream server will be saved on your PC in your Documents\Wheatstone\StreamBladeRemote\Backups folder. Files are saved in compressed .tar format, and are also uncompressed into clearly labeled folders for your convenience. If you do not have an archive program that can read a .tar, we recommend 7-Zip.



Includes all settings, licenses, presets and transform filters

Includes all logs and software update history

Includes stack traces and update history

DSP preset XML file

Lua metadata transform filter

Excludes: presets, filters, licenses, scheduler, system settings

Intended for full restoration after new SSD image

BACK-UP CONFIGURATION

Saves all settings, licenses, presets, and transform filters

BACK-UP LOGS

Saves all logs and software update history

Fetch Debug Diagnostics

In the unlikely event of a crash, this will gather some critical files off of the server into a tar file which can be sent to Wheatstone tech support to aid in understanding the cause of the crash.

This page will also give you the ability to upload DSP presets, Lua metadata transform filters, a configuration XML file, or an entire configuration from the factory from the remote PC.

Factory Reset

This function is enabled for the Admin user only. Resets the server to the state it was in when it left the factory, with the exception that the most recently installed channel license will not be removed. The config.tar file that is created from the Restore Entire Configuration function can be used to "unset" a factory reset.

APPLY SOFTWARE LICENSE

Your Wheatstream unit comes standard with three ingest channels. More channels may be added (up to eight) by contacting your sales representative and ordering a license. You will need a license seed for this, which you obtain by opening the SERVERS dialog and then clicking the UPDATE SERVER LICENSE button. The seed code for this device is displayed at the top. Copy and paste that into an email to your sales rep when you are ordering more channels. After obtaining a license file from Wheatstone, go back to this same dialog, and paste the license text into the License text area, then click UPDATE.

Click the UPDATE SERVER LICENSE button.

Paste the license text into the box or browse for the text file that has been provided. Apply.

UPDATE SERVER SOFTWARE

From time to time, Wheatstone will provide software updates for your Wheatstream device.

There are two types of update files: the Operating System (OS) and the Software (App).

The OS file is an update for the Linux operating system running on the Jetway PC. These are required only rarely.

The App file is an update for the Wheatstream application firmware. The vast majority of updates consist of a simple Application update only.

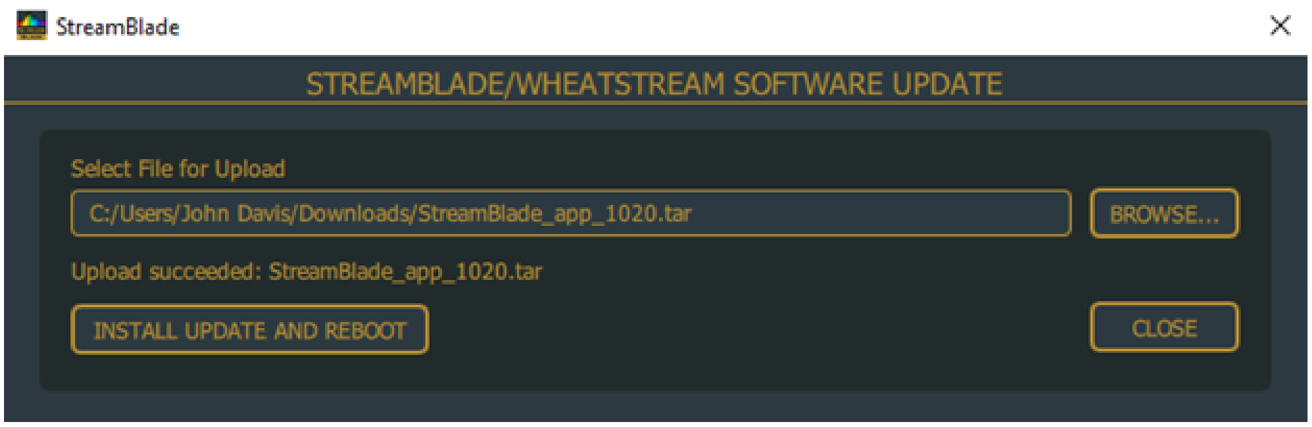
However, in the event that an OS update is required, it is critical that the OS update is performed first (before the Application update) in order to prevent locking up the Wheatstream device.

Both OS and App updates come in the form of "tar" files (a legacy Unix abbreviation for Tape ARchive). As of this writing (July 2023), the current OS image is R8 (StreamBlade_os_R8.tar), and is required for all versions of the application software from 1.0.1.016 onwards.

Note that application updates are often paired with an update of the Remote GUI. If this is the case, install the Windows Remote GUI first before proceeding with the Server update.

Updating Just The Application Software

1. Browse for the software update file (e.g. StreamBlade_app_2028.tar).
2. Click INSTALL UPDATE AND REBOOT.
3. The upload takes only a few seconds. You should see an “Upload Succeeded” message in both the Software Update dialog as well as in the Status pane.
4. The Software Update dialog will close after four seconds; however, the update itself takes about a minute and a half to install and then to reboot. The Remote GUI will not reconnect to the server by itself; you must reconnect manually. Once you are reconnected, confirm the new version number in the Main Menu pane.

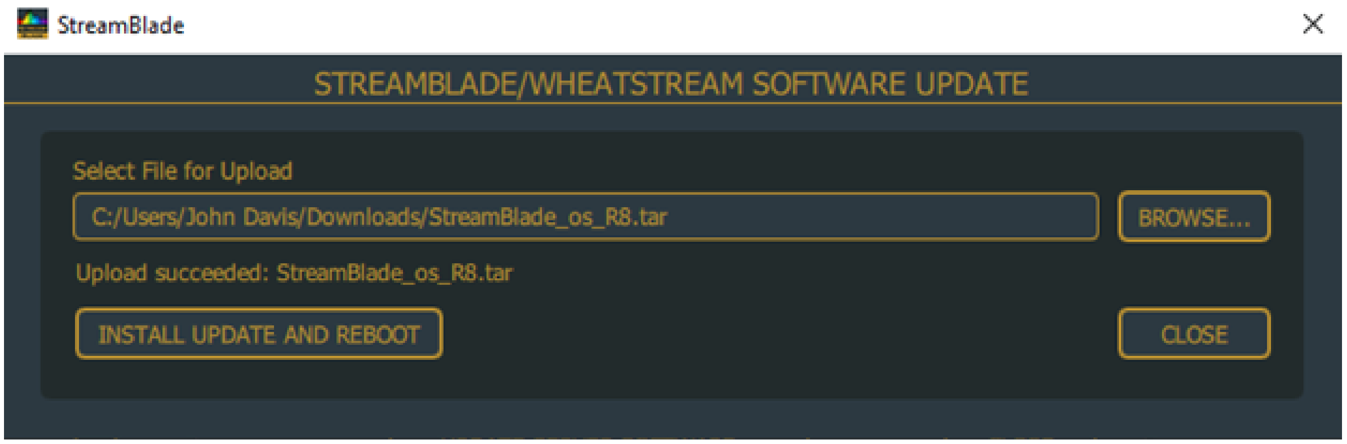

StreamBlade
STREAMBLADE/WHEATSTREAM SOFTWARE UPDATE
Select File for Upload
BROWSE...
INSTALL UPDATE AND REBOOT
CLOSE

C:/Users/John Davis/Downloads/StreamBlade_app_1020.tar

Upload succeeded: StreamBlade_app_1020.tar

Updating Both The OS and The Software.

1. Browse for the OS update file.
2. Click INSTALL UPDATE AND REBOOT.
3. The OS update file is significantly larger than an application update, so it will take several seconds to upload. You should see an “Upload Succeeded” message in both the Software Update dialog as well as in the Status pane once the upload completes.
4. The Software Update dialog will close after four seconds; however, the update itself takes about two minutes to install and then to reboot. The Remote GUI will not reconnect to the server by itself; you must reconnect manually. You will not see any explicit confirmation that the OS update has installed; however, the fact that the Wheatstream device is running and that you can connect to it is a strong indication that the update did succeed.
5. Now perform the application software update as described above.


StreamBlade

STREAMBLADE/WHEATSTREAM SOFTWARE UPDATE

Select File for Upload

C:/Users/John Davis/Downloads/StreamBlade_os_R8.tar

Upload succeeded: StreamBlade_os_R8.tar

ADVANCED CUSTOMIZATION

This section is intended for experienced Linux users and administrators.

Wheatstream runs on a single board computer running a version of the Debian 10 operating system. There is a single user, named `streamblade`, who resides at `/home/streamblade`. This is the location of most of the pertinent binaries and configuration files for Wheatstream.

This version of Debian 10 uses the `/etc/network/interfaces` file, along with the `ifup/ifdown` commands, to configure the network interfaces. If you look at the `interfaces` file you'll see that it sources another file called `network.cfg`, which you will find in the `/home/streamblade/config` folder. Changes you make to this file are persistent between reboots. Note that running the `ifdown/ifup` on `eth1` (the LAN interface) will cause Wheatstream to abort if it is running. It can be restarted with this command (from `/home/streamblade`):

```
./RunStreamBlade.sh &
```

If you need to customize DHCP behavior, you may create a custom `dhclient.conf` file and place it in `/home/streamblade/config`. If that file is present at boot-up, it is copied to `/etc/dhcp/dhclient.conf` and used during network startup. The default DHCP configuration is specified by the configuration found at `/home/streamblade/templates/dhconfig.h`.

Note that the only persistent changes you may make to the file system are to files residing in the `/home/streamblade/config` and `/home/streamblade/data` folders. Any other changes, anywhere on the file system, will be overwritten at next reboot.

If you require SSH/SCP access into Wheatstream, you may discover the password for the `streamblade` user by running this command:

```
/usr/local/bin/GenPW eth0
```

Warranty Statement

Limited Warranty by Wheatstone Corporation

1 All equipment sold and shipped to final destinations within the USA and its possessions warranted for one (1) full year from the date of purchase against defects in material and workmanship. All equipment sold and shipped to final destinations outside the U.S.A. and its possessions warranted for one (1) full year from the date of purchase against defects in material and workmanship. All repairs to maintain the unit at original specification will be made at no charge to the original purchaser, except for shipping and insurance costs to be prepaid by the owner to the factory in the event the unit cannot be serviced by an authorized Wheatstone Corporation dealer.

2 This Warranty is subject to the following restrictions and conditions:

a) The owner must have registered the product at Wheatstone's official web site; or at the time of servicing the owner must provide proof of purchase from an authorized Wheatstone Corporation sales engineer, distributor or dealer.

b) This Warranty is valid for the original purchaser on the unit. Parts used for replacement are warranted for the remainder of the original warranty period.Repair or replacement is in the discretion of Wheatstone Corporation and is the exclusive remedy hereunder.

c) This Warranty DOES NOT apply to damage or defects resulting from abuse, careless use, misuse, improper installation, electrical spikes or surges, or alteration, repair, or service of the unit or equipment by anyone other than Wheatstone Corporation or its authorized dealer.

d) This Warranty is void if the serial number has been removed, altered or defaced.

e) This Warranty DOES NOT cover loss or damage, direct or indirect, arising out of the use or inability to use this unit or for shipping or transportation to any dealer.

f) Wheatstone Corporation reserves the right to modify or change any unit in whole or in part at any time prior to return delivery in order to incorporate electronic or mechanical improvements deemed appropriate by the Wheatstone Corporation but without incurring any responsibility for modifications or changes of any unit previously delivered or to supply any new equipment in accordance with any earlier specifications.

g) THERE ARE NO OTHER WARRANTIES, EXPRESSED, IMPLIED, OR STATUTORY, INCLUDING ANY WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IF FOR ANY REASON ANY IMPLIED OR STATUTORY WARRANTY CANNOT BE DISCLAIMED, THEY ARE LIMITED TO THIRTY (30) DAYS FROM THE DATE OF PURCHASE. WHEATSTONE CORPORATION IS NOT RESPONSIBLE FOR ELECTRICAL DAMAGE, LOSS OF USE, INCONVENIENCE, DAMAGE TO OTHER PROPERTY, OR ANY OTHER INCIDENTAL OR CONSEQUENTIAL, WHETHER DIRECT OR INDIRECT, AND WHETHER ARISING IN CONTRACT, TORT, OR OTHERWISE. NO REPRESENTATIVES, DEALERS, OR WHEATSTONE PERSONNEL ARE AUTHORIZED TO MAKE ANY WARRANTIES, REPRESENTATIONS, OR GUARANTIES OTHER THAN THOSE EXPRESSLY STATED HEREIN.