

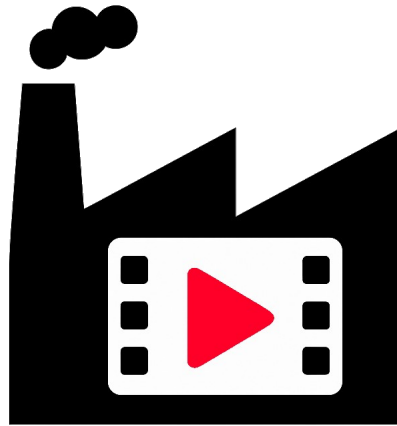
Free Factory QT

Documentation

by

James Hines and Karl Swisher

Licensed under the GPLv3 2013, 2025



FreeFactory

Introduction

Welcome to FreeFactory. FreeFactory not just another front end for ffmpeg, but one that fully encompasses the full and rich feature set of ffmpeg. While not everything ffmpeg can theoretically do is represented in the user interface, it allows for manual commands or even manual additions to what is selectable within the GUI making it extremely powerful without (many) cumbersome command lines.

FreeFactory has been around since 2012 and started its life as a collection of BASH scripts, with one watching over a “drop” folder waiting for files to arrive. When a file is dropped into a specified folder, the background process (FreeFactoryConversion) picks it up and processes according to the factory settings for that folder then saves it into a specific output folder.

These folders are specific to the type of output file you need after conversion and linked to a factory and typically look something like this:

/video/
/video/to-flac
/video/to-mxf-with-captions
/video/to-client-mp4
/video/dji-footage

In example, you have an audio file that needs converted to .flac, you simply drop one or more of the files into the to-flac folder and they are converted automatically for you to .flac and sent to whatever destination is assigned to that factory. The codec of the input file does not matter, as long as it is supported by ffmpeg, which most are. These folders can be local or network folders.

In 2013, FreeFactory got its first frontend in the form of a collection of tcl/tk scripts which made building “factories” much easier than doing it by hand. With the exception of getting updated whenever TCL or TK libraries changed and broke things and adding a manual options field, it pretty much remained in this state until recently.

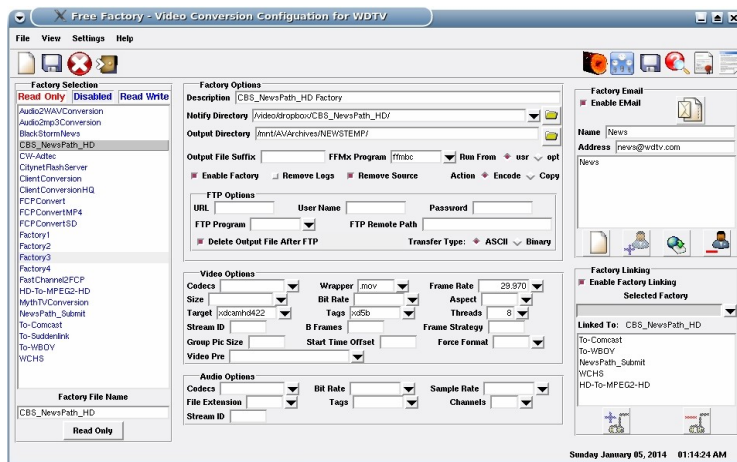


Figure 1: Original user interface using TCL-TK by Karl Swisher

Jump to 2025. I was needing to add some new features to FreeFactory for personal use and found that using TCL/TK a bit cumbersome and dated so we decided to rewrite it in Python3 using Qt6. This allowed for easy UI creation (using Qt6-Designer) and saved lots of time and code. In fact, because FreeFactoryQT uses a .ui file for the GUI, users can modify and re-arrange it to better suit their needs, if desired, without fear of breaking any python code. This assumes the user is familiar with the Qt6 Designer (Qt6) workflow.

While FreeFactory can use a background service process to pick up files from folders automatically (it’s now a systemd service) it also now supports direct drag and drop conversion without requiring the background process at all!

It also newly supports Streaming Factories with a built in Stream Manager allowing one to create and stream multiple streams. Of course streaming multiple streams is both CPU and/or GPU intensive so the amount of streams you can actually handle will greatly depend on the power of your hardware. Using your GPU is the way to go when doing this using codecs such as h264_nvenc and hevc_nvenc which leverage such hardware are good examples. Also, if you run out of NVEC co-processors, FreeFactory will automatically switch to libx264 for rendering. The Live Stream Manager tab also supports recording your video streams, or direct recording without streaming via the Mode switch (Stream|Record|Record and Stream).

Terminology

- **Factory:** This is a small config file which the Factory Builder will generate for you. It contains all the parameters necessary to create a command structure ffmpeg can use. The default location for these Factories is in /opt/FreeFactory/Factories. This can be changed in the Global Settings of the program.

- **Streaming Factory:** This is the mostly the same as above but contains stream specific parameters that normal Factories do not.

- **Drop Folder:** You create drop folders for dropping media files into for a specific conversion Factory. These are ONLY used by the background service. They are not used at all for Direct Encoding.

- **Direct Encoding:** This means the FreeFactory front end will directly do the encoding itself without the need for the background service. It is a drag and drop function.

NOTE: Needs more added

Manual Installation:

1. Download the .zip file from github.
2. Create a folder in /opt called FreeFactory (it is case sensitive)
3. Decompress the contents of the .zip file into /opt/FreeFactory/
4. (Optional) Create a desktop icon using the /opt/FreeFactory/Pics/FreeFactoryQT.ico file. For the shortcut to work, add the command: `python3 /opt/FreeFactory/bin/main.py`

First Run and Configuring FreeFactory

Requirements:

Python3

PyQt6

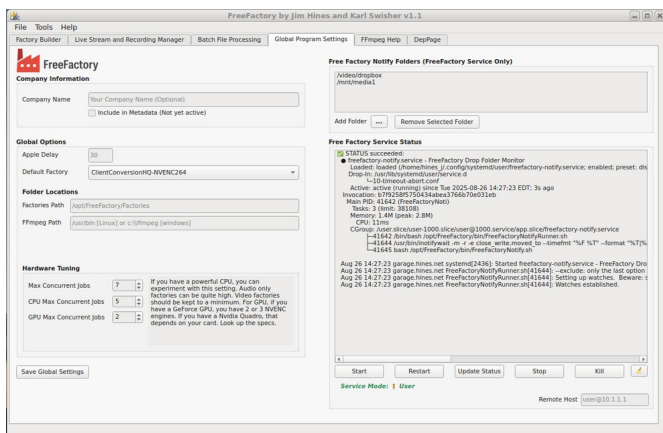
inotifywait (if using the background service)

Do not attempt to run FreeFactory as root user. Only the service should be ran as root (if needed).

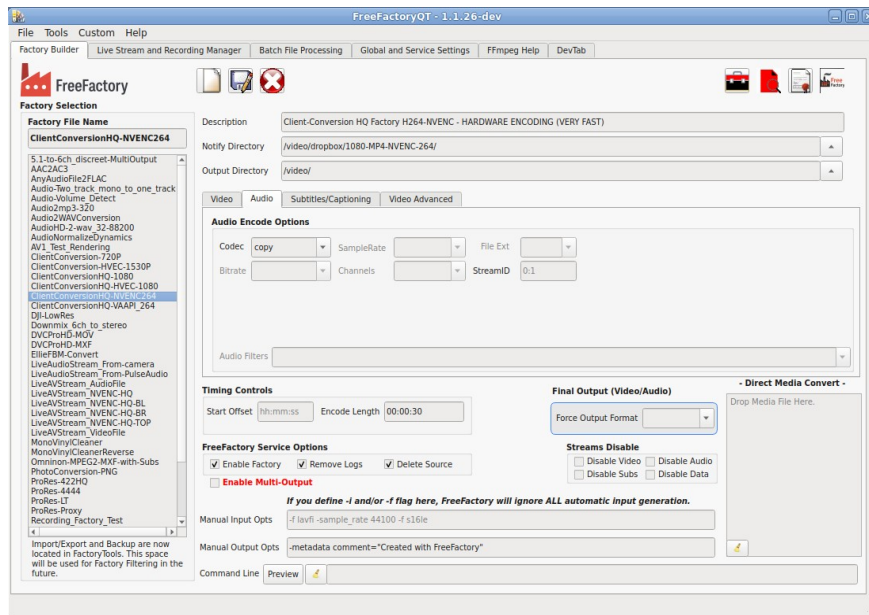
First, go to the Global Program Settings tab and simply click Save Settings on the bottom left. This ensures that ~/.freefactoryrc gets created in your HOME directory. The defaults here are fine for now for Linux installations.

You may fill in the Company Name if applicable and choose a default Factory here. This is the factory that you will commonly use the most for both Direct and Batch Encoding, if needed. This default Factory is for local encoding from FreeFactory and has absolutely nothing to do with the FreeFactoryNotify.service. You can also change your Factories folder and the PATH to your ffmpeg install if different than /usr/bin/.

You can also manage the FreeFactoryNotify.service from this tab.



Factory Builder Tab



TIP: The radio boxes not only can be used to select items from the built-in list, you may also type in anything manually. I.e. You may want to use a codec that is not in the built-in codec list or a custom bitrate that is not available in the combo list.

New Factory: This clears all fields in preparation for building a new factory.

Save Factory: This saves the current factory to filename.

Delete Factory: Deletes the currently selected factory.

Log Viewer: Views logs generated by the FreeFactory-Service.

License viewer: Views the GPLv3 license

About FreeFactory:

Factory List: Contains a list of all your factories.

Factory Filename: Name your factory here.

Factory Description: Add a description for your factory here. Be descriptive as once you get a lot of factories, it can avoid confusion.

Notify Directory: If you are only planning to use FreeFactory for direct conversion, then you can ignore this field completely. This will not apply to you. This is only used by the FreeFactoryConversion program which is ran by FreeFactory-Notify.service. If you plan on using the Drop Folders, this field is a REQUIREMENT.

Output Directory: This is the folder the factory will send you encoded file(s) to. It is used by both the GUI and the background FreeFactory service.

Video Tab:

The screenshot shows the 'Video Encode Options' tab in a video conversion application. The interface includes several dropdown menus and checkboxes for configuring video encoding. The 'Codec' is set to 'mpeg2video', 'Muxer' to '.mxh', and 'Frame Rate' to '30000/1001'. The 'Size' field is empty, 'Profile' is empty, and 'Profile Level' is empty. The 'Bitrate' is set to '25M', 'Pix_Fmt' to 'yuv422p', and 'Aspect' to '16:9'. The 'Lock Min/Max Bitrate' checkbox is checked. The 'Presets' dropdown is empty, and 'StreamID' is set to '0:0'. At the bottom, there is a 'Video Filters' dropdown menu.

Video Codec: If the file you are converting is a video file, you must specify a Codec here. Select copy if you want to keep the original codec and simply change the Wrapper/Muxer. As long as the codec is compatible with the wrapper, no other options are necessary*. Any fields left empty will simply default to the input file's settings except for this one. If the codec you need is not in the predefined list, you may type it in manually here. (command line option is -c:v)

***IMPORTANT:** When selecting "copy" in the codec widget, all video options requiring rendering will be blanked and ghosted preventing one from entering invalid options. This is because the codec "copy" is exclusive. If for example you would happen to have copy as the codec and a bitrate specified, copy is then ignored in favor of rendering because there is a bitrate specified. In this particular case, the original codec would be still maintained, but re-rendering would be forced with a new bitrate.

Video Size: Adjusts the output size or video resolution of the output file. If the size you need is not in the predefined list, you may type it in manually here. (command line option is -s)

Pix_Fmt (pixel format): This is the pixel format used for the codec. Not all pixel formats are compatible with all codecs. Use the FFmpeg Help tab for more info on specific pixel formats. You may also type in manually here if the pixel format you need is not listed. (command line option is -pix_fmt)

Muxer (Wrapper): This will tell FFmpeg the which wrapper to use for the file type expected at the output.

Profile:

Profile Level:

Video Bitrate: This sets the bitrate of the video encoding. (command line option is -b:v)

Use Min/Max Bitrate: This locks the minrate and maxrate values and includes these fields in the FFmpeg command. e.g. -b:v 50M -minrate 50M -maxrate 50M. Mostly needed only in professional formats.

Video Frame Rate: Sets the framerate for encoding. If left empty, it will use the frame rate of the input file. (command line option is -r)

Video Aspect Ratio: Sets the video aspect ratio. If left empty, it will use the aspect of the input file. (command line option is -aspect)

Video Presets: Selects a Video Preset. Command line option is -preset:v

Video Stream ID: Sets the stream id for the video stream. (command line option is -streamid:v)

Audio Tab:

The screenshot shows the 'Audio Encode Options' tab. It contains several input fields: 'Codec' is a dropdown menu with 'copy' selected; 'SampleRate' is a dropdown menu; 'File Ext' is a dropdown menu; 'Bitrate' is a dropdown menu; 'Channels' is a dropdown menu; and 'StreamID' is a text box with '0:1' entered. At the bottom, there is an 'Audio Filters' section with a large empty text area and a dropdown arrow on the right.

Audio Codec: If the file you are converting is an audio file, you must specify a codec here. Select copy if you want to keep the original codec*. Any fields left empty will simply default to the input file's settings except for this one. If the codec you need is not in the predefined list, you may type it in manually here. (command line option is -c:a)

***IMPORTANT:** When selecting “copy” in the audio codec widget, all options requiring rendering audio will be blanked and ghosted preventing one from entering invalid options. This is because the codec “copy” is exclusive. If for example you would happen to have copy as the codec and a bitrate specified, copy is then ignored in favor of rendering because there is a bitrate specified. In this particular case, the original codec would be still maintained, but re-rendering would be forced with a new bitrate.

Audio File Ext: Only useful if you are converting audio only files. **Use ONLY with pure AUDIO Factories.**

Audio Bitrate: Sets the bitrate of the audio stream or file. (command line option is -b:a)

Audio Samplerate: Sets the samplerate of the audio stream or file. (command line option is -ar)

Audio Channels: Sets the amount of audio channels of the audio stream or file. (command line option is -ac)

Audio Stream ID: Sets the stream ID of the audio stream when used with other audio or video streams. (command line option is -streamid:a)

Subtitles/Captions Tab:

The screenshot shows the 'Subtitle/Captioning Options' tab. It contains several input fields: 'Subtitle Codec' is a dropdown menu with 'copy' selected; 'Remove A53cc' is a checkbox; 'Sidecar frame rate' is a dropdown menu; 'Roll-up lines' is a text box with '3' entered; and 'Embed 608 as ST 436 (experimental)' is a checkbox.

Subtitle Codec: Selects the subtitle codec type. (command line option is -c:s). These can be copy, ass, mov_text, srt, ssa, Sidecar TTML and Sidecar SCC. The Sidecar selections are special use cases for playback devices which support Sidecar subtitles, usually only used in professional environments. Also supported by VLC in many cases.

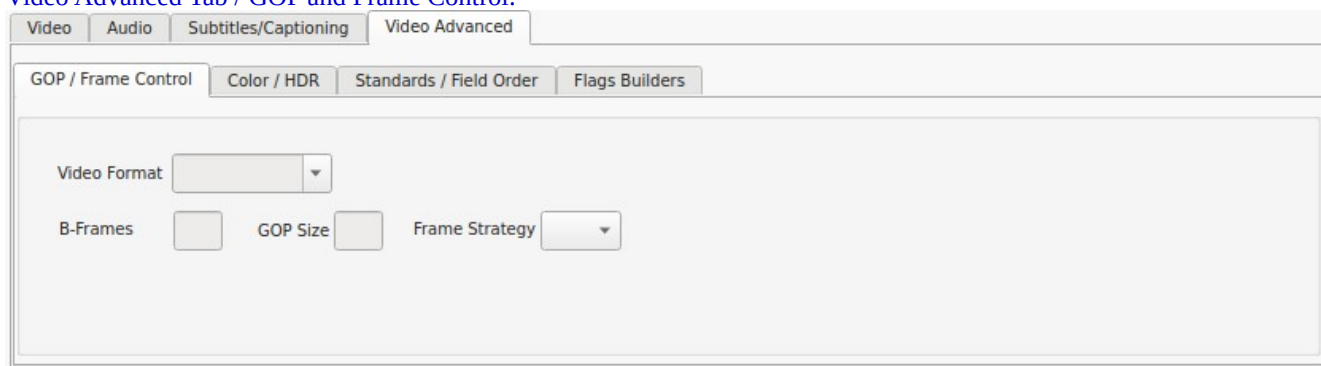
RemoveA53cc: If checked, will remove DTVCC from the video stream (-a53cc 0). The default behavior of FFmpeg is to always pass these types of subtitles even when re-rendering video.

Sidecar Frame Rate: Sets the framerate of SCC subtitles.

Roll-Up Lines: Sets how lines will roll up during captioning of SCC subtitles.

Embed 608 as ST436: (Introduced in FFmpeg v8.x). This adds the bitstream filter eia_to_smpte436m to the encode and will add eia608 subtitles as ST436 data.

Video Advanced Tab / GOP and Frame Control:



The screenshot shows the 'Video Advanced' tab with the 'GOP / Frame Control' sub-tab selected. The settings include: 'Video Format' (a dropdown menu), 'B-Frames' (a checkbox), 'GOP Size' (a text input field), and 'Frame Strategy' (a dropdown menu).

Video Format: Forces the video format for component, pal, ntsc, secam and mac. Generally for professional use and not normally need for casual users.

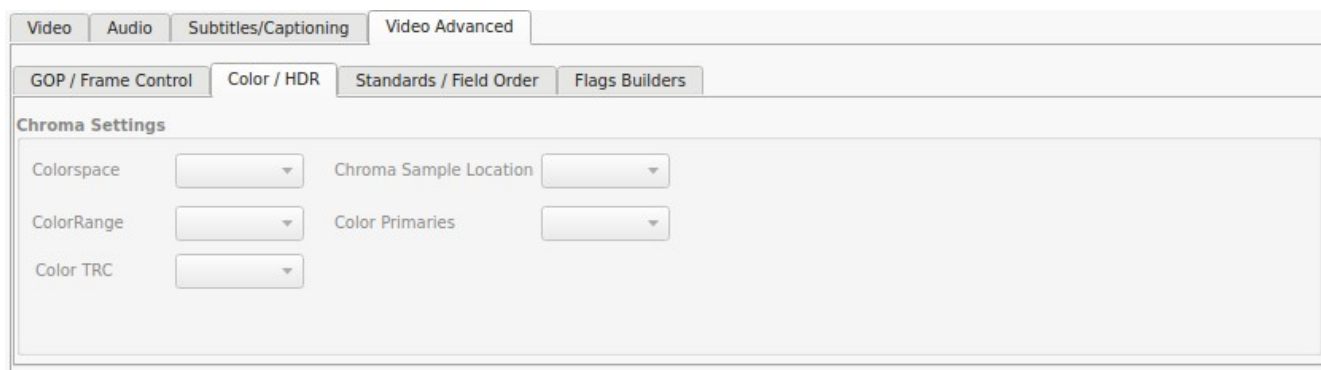
Video B-Frames: Sets the bi-directional frames. (command line option is -bf)

GOP Size: Sets the Group of Pictures size. (command line option is -g)

Frame Strategy:

Video Advanced Tab / Color and HDR:

These advanced color settings are only useful for some codecs. Mpeg2video and h26x are good examples.



The screenshot shows the 'Video Advanced' tab with the 'Color / HDR' sub-tab selected. The settings are grouped under 'Chroma Settings' and include: 'Colorspace' (dropdown), 'Chroma Sample Location' (dropdown), 'ColorRange' (dropdown), 'Color Primaries' (dropdown), and 'Color TRC' (dropdown).

Colorspace: Sets the color space.

Colorrange: Sets the color range.

ColorTRC: Selects color transfer.

Chroma Sample Location: Choose the chroma sample location.

Color Primaries: Sets color primaries.

Video Advanced Tab / Standards and Field Order:

The screenshot shows the 'Video Advanced' tab with the 'Standards / Field Order' sub-tab selected. It contains two main sections: 'GOP / Frame Control' and 'Quantizer / Misc'. In 'GOP / Frame Control', there are checkboxes for 'Alternate Scan' and 'Non Linear Quant', and dropdown menus for 'Signal Standard', 'Seq Disp Ext', and 'Field Order'. In 'Quantizer / Misc', there is a checkbox for 'Intra VLC', a 'DC' value of -8, an 'Enable' checkbox, 'QMin' and 'QMax' values of -1, 'RC Init Occupancy' of 17M, and 'BufSize' of 17M.

Alternate Scan: Enables the alternate scantable.

Non Linear Quantize: Uses nonlinear quantizer.

Signal Standard: Force/set Signal Standard (mxf files).

Seq Disp Ext: Writes sequence display extension blocks.

Field Order: Change field order on interlaced files.

Intra VLC: Use MPEG-2 intra VLC table.

DC: Intra_dc_precision from -8 to 16.

Video Advanced Tab / Flags Builders:

The screenshot shows the 'Video Advanced' tab with the 'Flags Builders' sub-tab selected. It contains two main sections: 'Flags Builder' and 'Flags2 Builder'. 'Flags Builder' has checkboxes for 'unaligned', 'ilme' (checked), 'qpel', 'loop', 'gray', 'psnr', 'ildct' (checked), 'low_delay', 'aic', 'cgop', 'global_header', 'bitexact', 'output_corrupt', and 'mv4'. Below these is a text box containing '+ildct+ilme'. 'Flags2 Builder' has checkboxes for 'fast', 'noout', 'ignorecrop', 'local_header', 'chunks', 'ass_ro_flush_noop', 'export_mvs', 'skip_manual', 'showall', and 'icc_profiles'. Below these is an empty text box.

Flags Builder: Advanced encoder bitstream flags (-flags). Optional; most encodes don't need these. Checked items are collected into FLAGS as +tokens. Disabled when Video Codec=copy.

Flags2 Builder: Additional/legacy flags (-flags2). Mostly for analysis or bitstream quirks. Checked items are collected into FLAGS2 as +tokens. Often ignored by some encoders.

Bottom of Factory Builder Tab:

Timing Controls:

Start Offset: This tells FFmpeg where to start encoding within the file, in frames. (command line option is -ss)

Encode Length: This is the length to encode. This is very useful for testing factories for long form programming. (command line option is -t)

FreeFactory Service Options:

Enable Factory: (Default Enabled) Only used with the FreeFactory service. Disabled Factories are ignored by the service.

Remove Logs: (Default Enabled) Only used with the FreeFactory service. This forces FreeFactory service to remove logs after completely a job. If you are experiencing encoding issues, turn this off so you may view the log which is saved in /var/log/FreeFactory.

Delete Source: (Default Enabled) Only used with the FreeFactory service. Deletes the original file from the drop folder once processing has completed successfully.

Final Output:

Force Format: Forces the format of the output file. ie. can be used to force a mp4 file to flv for streaming. (command line option is -f)

Enable Multi-Output: When enabled, this will take FreeFactory out of single file generation mode and allow multiple output files to be created. This is useful for splitting up a 5.1 channel audio stream into six individual audio files. This can also be used to split a media file into separate streams such as .m4v, .ac3, .srt, etc. When selected, options that pertain to single file output are ghosted and unusable by default such as Muxer, Notify Directory, Force Output Format, Audio File Ext and Subtitle codec.

Each output must be specified for OUTDIR and STEM({outdir}{stem}). OUTDIR is pulled from the Output Directory field and STEM is pulled from the dropped filename. ***This advanced feature only works with local drag and drop encoding and does not work with the FreeFactory Service.***

Streams Disable: There are four options here. Disable Video, Audio, Subs and Data. This attempts on a per stream basis to ignore and not pass these types of streams. Disable Video is very useful when converting audio files that may have embedded covers, video clips or photos. Disable Subs and/or Data to remove lyrics and other (useless) data.

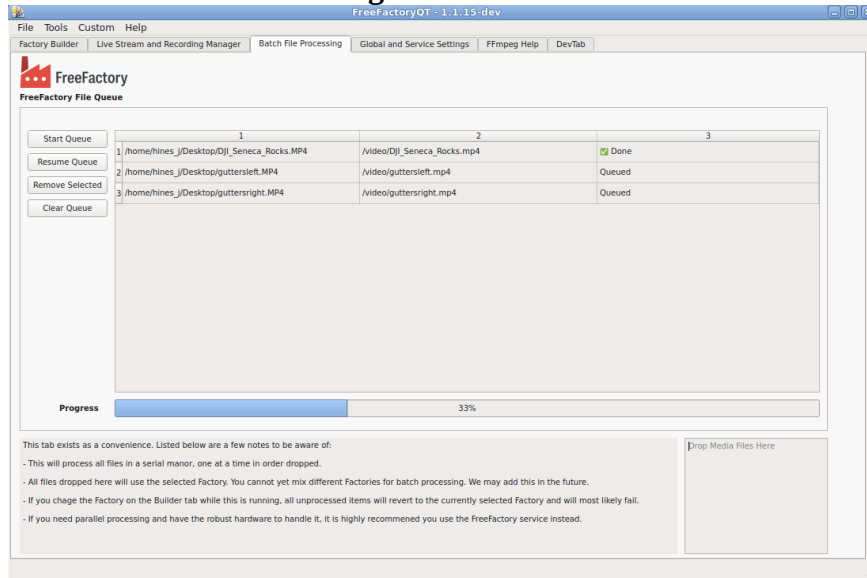
Manual Input Options: These are input options only. This field (and Manual Output Opts) is the REAL power of FreeFactory. Options in this field always end up BEFORE the -i [filename]. Input options can include -f, -i or even hardware options for Intel CPUs (-hwaccel vaapi -hwaccel_device intel).

Manual Output Options: These are output only. This field is the REAL power of FreeFactory. While FreeFactory attempts to provide the most used options in the GUI, there is simply no way it can because there are so many. Advanced users can use this field to build complete FFmpeg command line structures, or just add options to your GUI options that are unavailable otherwise as well.

Command Line Preview: This allows you to see the complete FFmpeg command that FreeFactory generates. It combines all the UI selections as well as the Manual Options and Streaming Options. You can also copy this line and try it from your shell to test for issues.

Drop Zone: This allows dropping a file to be converted using the currently selected Factory. Combined with Encode Length, this is a great way to test factories, especially when creating factories for the background service.

Local Batch File Processing Tab



This tab is used for processing multiple files. Drop your files into the (#6) Drop Zone. The files are encoded using the currently selected factory in the Factory Builder tab. Do NOT change Factories on the Builder Tab while batch processing files or the newly selected Factory will be applied to the next item in the Queue table. The files will be output to the Output Directory specified in your Factory.

Start Queue: Starts encoding the file list.

Pause/Resume Queue: Does just what it says. Pauses and resumes encoding. *If a file is encoding when you click Pause, it will continue to render until finishing or fail and then Pause the queue.*

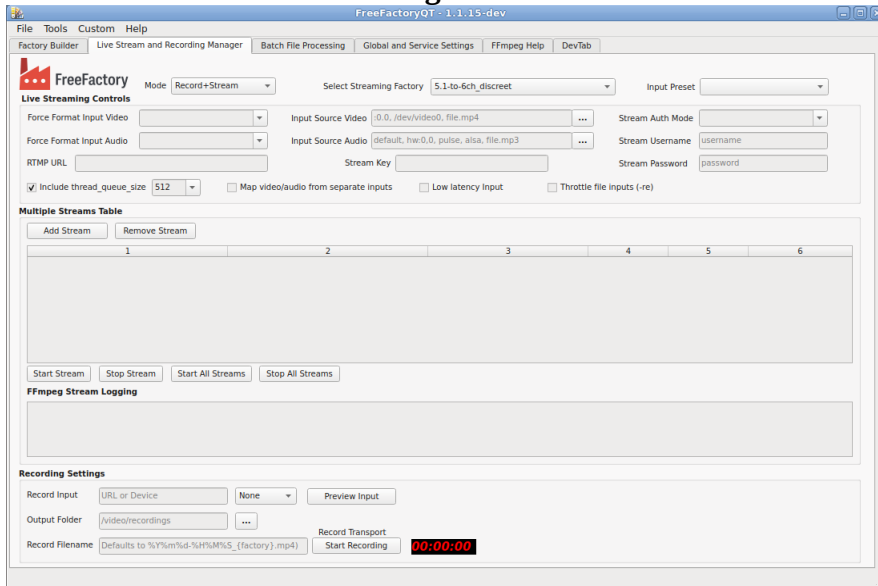
Remove Selected: Removes selected items from the Queue table.

Clear Queue: Clears the Queue table.

Queue Table: The list of files to be encoded.

File Drop Zone: Drop your files here.

Live Stream and Record Manager tab



Mode: This button selects the tab mode which can be one of Off, Stream, Record or Record+Stream. Selecting Stream mode will unlock all the Streaming Controls. Selecting Record mode will unlock the recording widgets. Selecting Record+Stream will unlock the entire tab. This selection gets saved as a Factory Key (STREAMMGRMODE) and is used to identify the factory type.

Select Streaming Factory: This now follows the factory selection from the Builder tab if the factory selected has mode set to anything but “Off”. You should never need to click on this as it does nothing except show you the selected stream or recording factory. It is now filtered to only contain Stream and Record factories. You must first select your Stream Factory from the Factory Builder tab, switch back to this tab then click **Add Stream** to add it to the Stream Table. You must repeat this process for each Streaming Factory added. If you make any edits, you can use the hotkey combination CTRL+S to save the factory. This prevents you from having to go back and forth between the Builder and Streaming tabs just to save changes.

Input Preset: (not yet active) This will eventually be used to create basic beginner point presets.

Force Format Input Video: This can be any of file, x11grab, xcbgrab, v4l2, lavfi or concat.

Force Format Input Audio: This can be any of file, pulse, alsa, dshow or concat.

Input Source Video: This is a screen (:0.0), a video device or a file.

Input Source Audio: This can be default, hw:n,n, pulse, alsa, etc.

Stream Key: Unique stream key for your account or a unique string if you have your own server.

Add Stream: Adds a stream to the Stream Table

Remove Stream: Removes a stream to the Stream Table.

Start Stream: Starts the Selected stream.

Stop Stream: Stops the Selected stream.

Start All Streams: Starts ALL streams.

Stop All Streams: Stops ALL streams.

FFmpeg Stream Logging: Shows the output of FFmpeg during encoding/streaming.

RTMP URL: RTMP URL.

Record Input:

Output Folder:

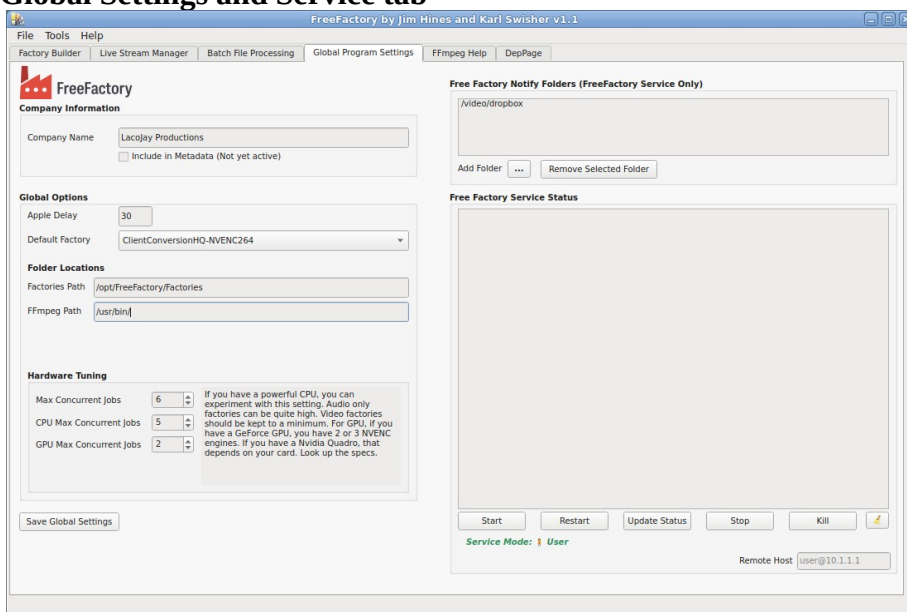
Record Filename:

Recording Input Selector:

Preview Input:

Start/Stop Recording (Toggle):

Global Settings and Service tab



Company Name: (Optional) Enter your company name here.

Include in Metadata: This will include your company name in all metadata of converted files (Not yet enabled).

File Copy Delay (Previously Apple Delay): This is deprecated and left in for backwards compatibility.

Default Factory: Chooses a default Factory that is loaded whenever FreeFactoryQT is started (optional).

Factories Path: This allows users to move the /opt/FreeFactory/Factories folder to a different location or switch between multiple folders.

FFmpeg Path: Defaults to /usr/bin but can be changed to ie. /usr/local/bin.

Hardware Tuning/Max Concurrent Jobs: This sets the maximum number of rendering jobs that can be processed in parallel. Add together the Max GPU + CPU and set this number to the total.

Hardware Tuning/CPU Max Concurrent Jobs: This is the maximum CPU limit. If you only process audio files, this can be quite high.

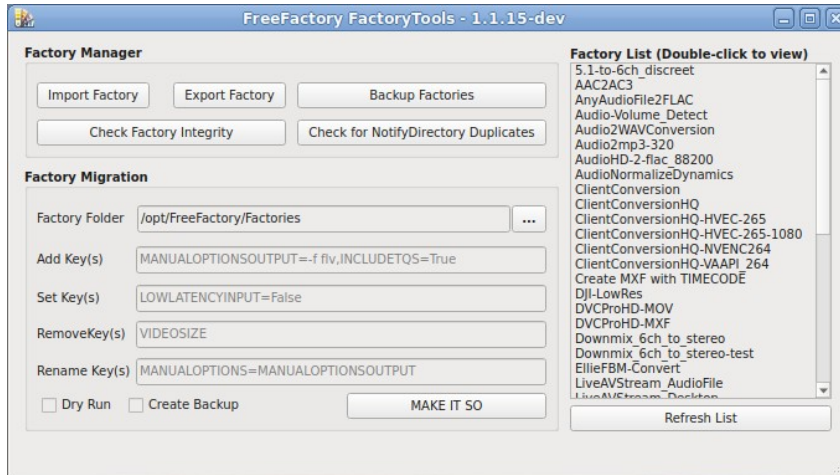
Hardware Tuning/GPU Max Concurrent Jobs: This sets the maximum GPU limit. For GeForce cards using NVENC, this usually can be set to 2 or 3. If you have a Quadro, check your card documentation.

Free Factory Notify Folders: Allows multiple watched folders. Useful for network shares.

Clear: Clears the Status Window.

This tab is not for general use and is only a place holder for experimental/deprecated widgets. Please do not use. It will be removed once we get to a stable release.

FreeFactory Factory Tools App



The Factory Tools app allows for Factory Management. This allows you to Import/Export and Backup your Factories. It also does limited integrity testing and checks for duplicate Notify folders (these all have to be unique).

Factory Manager Section:

Import Factory: Clicking this will allow you to import a Factory. You may multi-selected imported files. If you import a file that has the same name as one you already have, it will get a new suffix and not overwrite your existing factory.

Export Factory: This will allow exporting factories either as flat files or a zipped file. You may select multiple factories for exporting. As a security feature, when exporting, all PATHS will be cleared as well as any passwords and streaming paths/filenames. Exported factories by default will be saved to the `/opt/FreeFactory/export` folder. You can of course change this to where ever you like.

Backup Factories: This will backup your entire Factories folder as a .zip file.

Check Factory Integrity: This checks the Factory for malformed keys.

Check for NotifyDirectory Dups: This will scan your entire Factories folder and check for any duplicate Notify Folders. Each one must be unique.

Factory Migration Section:

Factory Folder: Select the Factory Folder you wish to use. Usually `/opt/FreeFactory/Factories` but this is optional.

Add Key(s): Adds new Factory Keys to the Factory File. Ie: `MANUALOPTIONSOUTPUT,NEWKEY2,ETC`

Set Keys(s): Sets an already existing Key to a value. Ie: `MANUALOPTIONSOUTPUT=-comment "Created with FreeFactory"`

Remove Key(s): Removes the Key from the Factory.

Rename Key(s): Renames the Factory Key.

Dry Run: Will show you what will be done without actually doing anything.

Create Backup: Will first backup all your factories by appending a .bak at the end, before performing any actual operations.

Make it So: Runs the job.

Factory List: Displays your current and selected Factory folder. You may multi-select from this list when doing Exports. You can also double click a Factory to preview it (but you cannot edit it).

What the heck is the FreeFactory service and do I need it?

The FreeFactory service is mostly useful for professionals and power users with multiple users and multiple “drop” folders, one per Factory. Depending on which folder a file is dropped into, the file gets picked up and converted according to the settings of one factory with a Notify folder assigned. Each folder can have only one unique Notify folder assigned to it (you can check for duplicates using the newly added Factory Tools app). This is a simple example of a useful directory tree for drop folders:

```
# tree /video
/video
├── 844938.mxf
├── dropbox
│   ├── 1080-MP4
│   ├── 1080-MP4-HEVC-265
│   ├── 1080-MP4-NVENC-264
│   ├── 720P-MP4
│   ├── Audio2FLAC
│   ├── Audio2MP3-320
│   ├── Audio2Wav
│   ├── photos
│   ├── Testing
│   ├── To-MKV
│   └── WDTV-1
└── Track07.mp3
```

Of course the directory permissions must be set correctly for all users to have access to which folders they will need, and also the output folders as well. All these are configured in the Factory Builder tab. Many of the included example Factories will use one of these folders as the Notify Directory and the /video root for the destination.

The FreeFactory-notify service runs a “watch” script (FreeFactoryNotifyRunner.sh) which uses inotifywait to watch all these folders within the /video folder. Whenever a file is dropped or copied into a watched folder, the script then runs FreeFactoryConversion.tcl and encodes the file according the Factory that is assigned to the folder. Since this completely runs in the background, it can be left running just as any other service. When using the FreeFactory-notify service, there is no need to have the FreeFactoryQT front end running at all unless you are building and/or testing new Factories.

If you are indeed allowing multiple users to access these drop folders, then you should run the service system-side, as root. This is one of the options provided when running the service installer script. You can still control the (root) service as a user, but it will ask for a sudo password if you need to control it.

You can also control the service using the normal systemctl program.

This is the order the service launches a conversion job:

```
FreeFactory-notify.service >> FreeFactoryNotifyRunner.sh >> FreeFactoryNotify.sh >> FreeFactoryConversion.tcl
```

IMPORTANT: The FreeFactoryConversion.py program will process all dropped files in parallel launching a new FFmpeg instance for each file dropped. If you expect a lot of files being dropped, either have a very beefy server with lots of memory and a capable CPU/GPU for handling lots of rendering traffic, or don’t drop too many at once. This will vary from system to system. Dropping more files than your system can handle will create very long encode times, or worse, crash your rendering server.

EDIT: The above has been negated by the addition of the Hardware Tuning section above.


Setting up the FreeFactory Service:



Setting up FreeFactory service:


This requires inotifywait installed on your system.

Simply run the script setup-notifyservice.sh.

```
$ ./setup-notifyservice.sh
```

 FreeFactory Notify Service Setup

 Checking if the service is currently running...
 Service is not currently running.

 Installed in USER mode

Choose an action:

- 1) Install/enable in USER mode
- 2) Install/enable in SYSTEM-WIDE mode (Requires root privileges via sudo)
- 3) Uninstall from USER mode
- 4) Uninstall from SYSTEM-WIDE mode (Requires root privileges via sudo)
- 5) Quit

Selection:

Unless you are running the FreeFactory service for a company environment, you will want to select USER mode.

Once setup, you can run:

```
systemctl --user start freefactory-notify.service  
and  
systemctl --user enable freefactory-notify.service  
to start whenever the user logs in.
```

You can also start and stop the service from within the FreeFactoryQT user interface.

Converting Files using Multiple Factories through Watched Drop Folders (Service Only)

Sometimes it's just not possible to end up with a file format you need with only one encoding pass. This can easily be done by simply setting the output folder of Factory "A" to the Watched (input) folder of Factory "B". Once Factory A completes the file, Factory B will then pick it up for additional processing and deliver to the output folder set in Factory B. This output folder can not be a watched folder. You can string as many factories together in this manner as you need, but more than two might become confusing.

One example use of this would be to convert a .wav file to a .flac with a different codec. e.g. Converting a 32bit .wav to a 16bit or 24bit .flac. Since flac is considered an audio codec in FFmpeg, you cannot also choose for instance codec pcm_s16le and directly convert it into a 16bit .flac file. You must convert it using the pcm_* codec first before creating the flac file.

Factory A would convert the 32bit input file to a pcm_s16le (or s24le) and deliver to a watched folder for creating FLAC files. Once the converted WAV is completed and arriving at Factory B's watched folder, it will again be picked up and converted to FLAC maintaining the previously rendered 16bit or 24bit file and compress it into a .flac file.

Included Sample Factories:

FreeFactory includes several pre-built factories to help get you up and running quickly and give you some idea of what is expected in a functional factory. Please use these as a learning guide or even modify them to be your daily use factories. You will need to change any Notify and Output folders to match YOUR system as there were all built for mine.

5.1-to-6ch_discrete-MultiOutput:

Factory Type: Local Encode Multi-Output

This factory will convert any multichannel audio stream and break it down to individual .wav files. This is a local encode factory and should not be used with the FreeFactory Notify Service. It will accept most video files that includes a multi-channel audio stream whether it be ACC, AC3 or DTS or from audio files (.ac3, aac, m4a, dts).

AAC2AC3:

Factory Type: Local Encode and FreeFactory Service

Drop a video file with anything but ac3 audio and it will give you a new file which has ac3 audio @320Mb/s. You should be safe to keep the Video Codec in copy mode. Change any other parameters to suit your needs.

AnyAudioFiletoFLAC:

Factory Type: Local Encode and FreeFactory Service

This will convert any audio file to a FLAC with compression level 8. It will also strip out any embedded photos which may be present via the Disable Video checkbox. If you wish to keep embedded photos, uncheck this box and re-save the factory.

Audio-Two_track_mono_to_one_track:

Factory Type: Local Encode (optional FreeFactoryService)

This will take any two channel mono file and convert to a one channel mono file thus saving disc space. Also strips embedded photos.

Audio-Volume_Detect:

Factory Type: Local Encode

This factory simply generates a list of peak volume levels for audio files. Not really useful within FreeFactory at the moment as there is no mechanism to print out the text.

Audio2mp3-320:

Factory Type: Local Encode and FreeFactory Service

This factory does exactly what the description says. Converts any audio file to 320Mb/s mp3 while stripping any embedded images.

Audio2WAVConversion:

Factory Type: Local Encode (optional FreeFactoryService)

This factory does exactly what the description says. Converts any audio file to a s16le wav file.

AudioHD-2-wav_32-88200:

Factory Type: Local Encode and FreeFactoryService

This factory was created to convert DSD audio files to 32bit wav files.

AudioNormalizeDynamics:

Factory Type: Local Encode

This factory is useful for cleaning up spoken voice dynamics. This will completely destroy music, but is great for spoken audio.

ClientConversion-720P:

Factory Type: Local Encode and FreeFactory Service

This was created to make 30 second television commercials email-able for client viewing. It does hardware encoding via the h264_nvenc codec @720P/1.5Mb/s.

Downmix_6ch_to_stereo:

Factory Type: Local Encode

This was created to convert multi-channel audio to stereo using only audio filters but still retaining the LFE channel (in the stereo mix).

DVCProHD-MOV:

Factory Type: Local Encode (optional FreeFactoryService)

This is a very basic DVCProHD factory. Please modify to suit your needs. Outputs in .mov format.

DVCProHD-MXF:

Factory Type: Local Encode (optional FreeFactoryService)

This is a very basic DVCProHD factory. Please modify to suit your needs. Outputs in .mxl format.

LiveAudioStream_From-PulseAudio:

Factory Type: Local Stream

Streaming example for Pulse audio.

LiveAVStream_AudioFile:

Factory Type: Local Stream

Streaming example for live streaming an audio file.

LiveAVStream_VideoFile:

Factory Type: Local Stream

Streaming example for live streaming a video file with audio.

ProRes-422HQ:

Factory Type: Local Encode (optional FreeFactoryService)

Create a ProRes 422HQ file using prores_ks codec and 24 bit audio.

ProRes-4444:

Factory Type: Local Encode (optional FreeFactoryService)

Create a ProRes 4444 file using prores_ks codec and 24 bit audio and yuv422p10le.

Recording_Factory_Test:

Factory Type: Recording Factory

Records a live stream to your HDD from a locally installed camera.