Free Factory QT

Documentation

by

Jim Hines and Karl Swisher
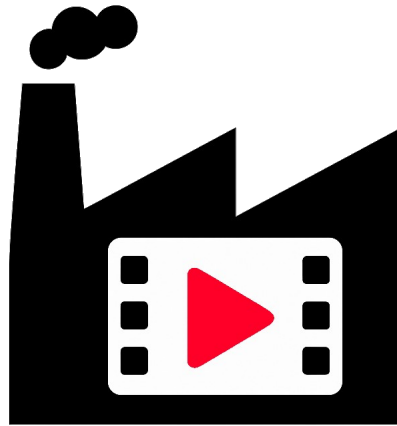
FreeFactory

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 supported by ffmpeg, which most are. These folders can be local or network folders.

In 2013, FreeFactory got it's 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.
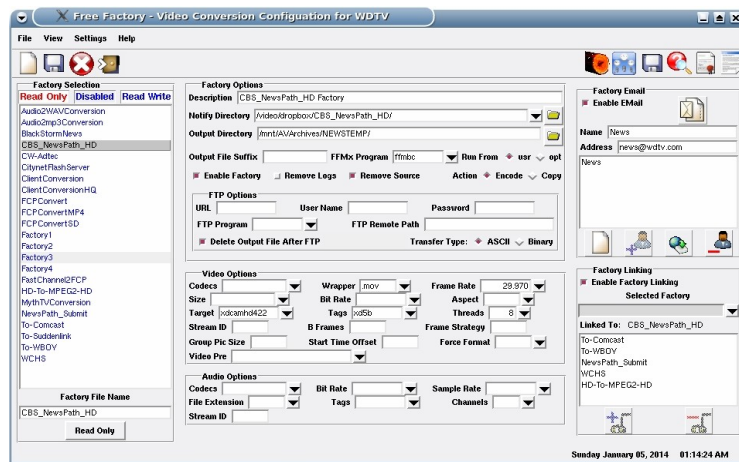

Figure 1: Original user interface using TCL-TK by Karl Swisher

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 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.

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
inotifiywait (if using the background service)
TCL (if using the background service)

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.
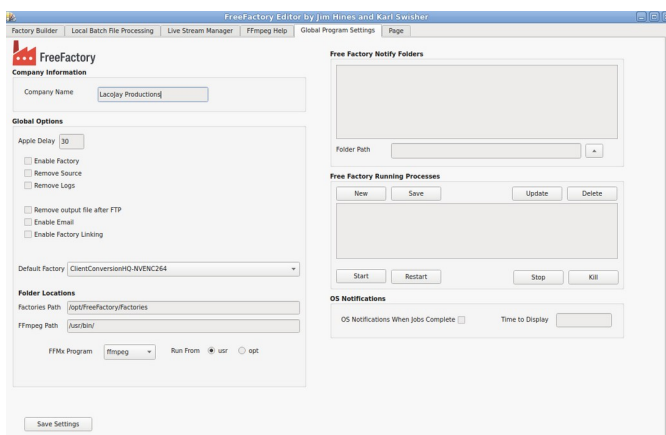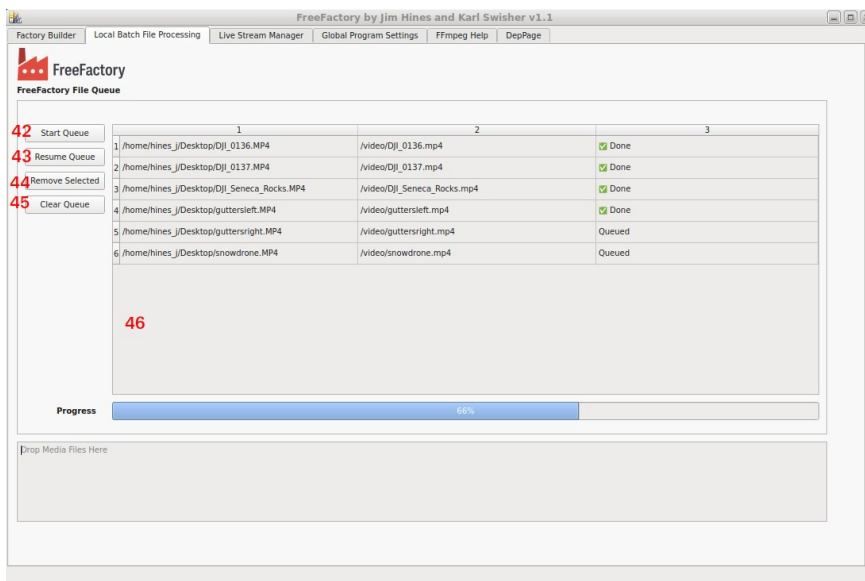
Factory Builder Tab
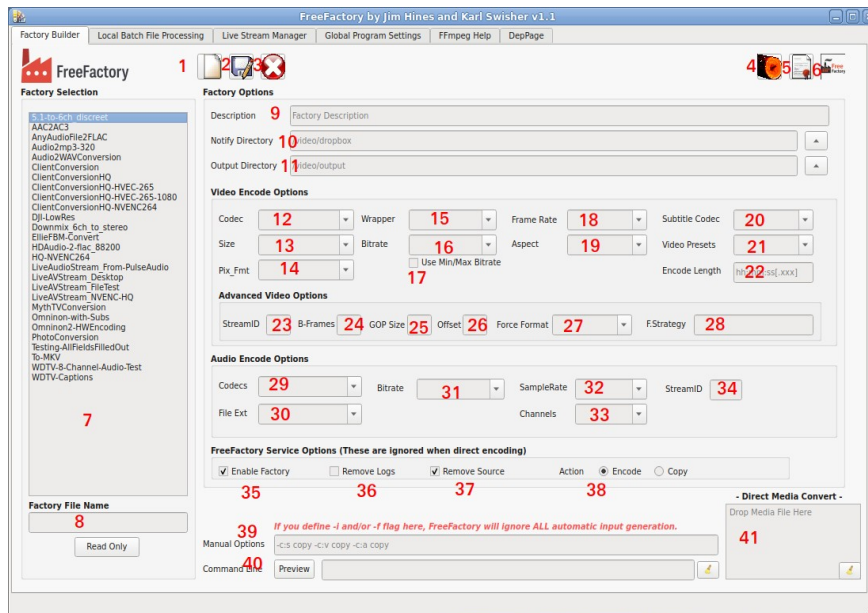
TIP: The radio boxes not only can be used to select items from the built-in list, you may also type in anything manually. I.e. You may want to use a codec that is not in the built-in codec list or a custom bitrate that is not available in the radio list. Example. You select 60M from the Video Bitrate radio box, but you want to add min and max as well. Just add this "-minrate 60M -maxrate 60M" after the 60M in the box and save the factory. This will be added the FFmpeg command string as -b:r 60M -minrate 60M -maxrate 60M.

1. New Factory: This clears all fields in preparation for building a new factory.
2. Save Factory: This saves the current factory to filename (see #8).
3. Delete Factory: Deletes the currently selected factory.
4. Log Viewer: Views logs generated by the FreeFactory-Service.
5. License viewer: Views the GPLv3 license
6. About FreeFactory:
7. Factory List: Contains a list of all your factories.
8. Factory Filename: Name your factory here.
9. Factory Description: Add a description for your factory here. Be descriptive as once you get a lot of factories, it can avoid confusion.
10. 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.
11. 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.
12. 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. 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)
13. 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)
14. Pix_Fmt: 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 pix_fmt you need is not listed. (command line option is -pix_fmt)
15. Wrapper: This will tell FFmpeg the which wrapper to use for the file type expected at the output.
16. Video Bitrate: This sets the bitrate of the video encoding. (command line option is -b:v)
17. Use Min/Max Bitrate: This locks the minbitrate and maxbitrate values and includes these fields in the FFmpg command. ie -b:v 50M -minbitrate 50M -maxbitrate 50M

18. 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)
19. 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)
20. Subtitle Codec: Selects the subtitle codec type. (command line option is -c:s)
21. Video Presets: Selects a Video Preset. Usually used for live stream encoding. (command line option is -preset)
22. Encode Length: This is the length to encode. This is very useful for testing factories for long form programming. (command line option is -t)
23. Video Stream ID: Sets the stream id for the video stream. (command line option is -streamid:v)
24. Video B-Frames: Sets the bi-directional frames. (command line option is -bf)
25. GOP Size: Sets the Group of Pictures size. (command line option is -g)
26. Start Offset: This tells FFmpeg where to start encoding within the file, in frames. (command line option is -ss)
27. 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)
28. Frame Strategy: Deprecated. Will be removed.
29. 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)
30. Audio File Extension: Only useful if you are converting audio files. This basically lets FFmpeg decide what to do.
31. Audio Bitrate: Sets the bitrate of the audio stream or file. (command line option is -b:a)
32 Audio Samplerate: Sets the samplerate of the audio stream or file. (command line option is -ar)
33. Audio Channels: Sets the amount of audio channels of the audio stream or file. (command line option is -ac)
34. 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)
35. Enable Factory: Only used with the FreeFactory service. Disabled Factories are ignored by the service.
36. Remove Logs: Only used with the FreeFactory service.
37. Remove Source: Only used with the FreeFactory service.
38. Action: Deprecated and needs removed. This is a leftover from the original FreeFactory TCL app and left in for Factory compatibility.
39. Manual 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.
NN. Manual In Options (Not Pictured): This allows one to enter manual options that come BEFORE the input file (-i).
40. 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.
41. Drop Zone: This allows dropping a file to be converted using the currently selected Factory.

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.

42. Start Queue: Starts encoding the file list.
43. Pause/Resume Queue: Does just what it says. Pauses and resumes encoding.
44. Remove Selected: Removes selected items from the Queue table.
44. Clear Queue: Clears all items in the Queue table.
46. Queue Table: The list of files to be encoded.
47. File Drop Zone: Drop your files here.

Live Stream Manager tab
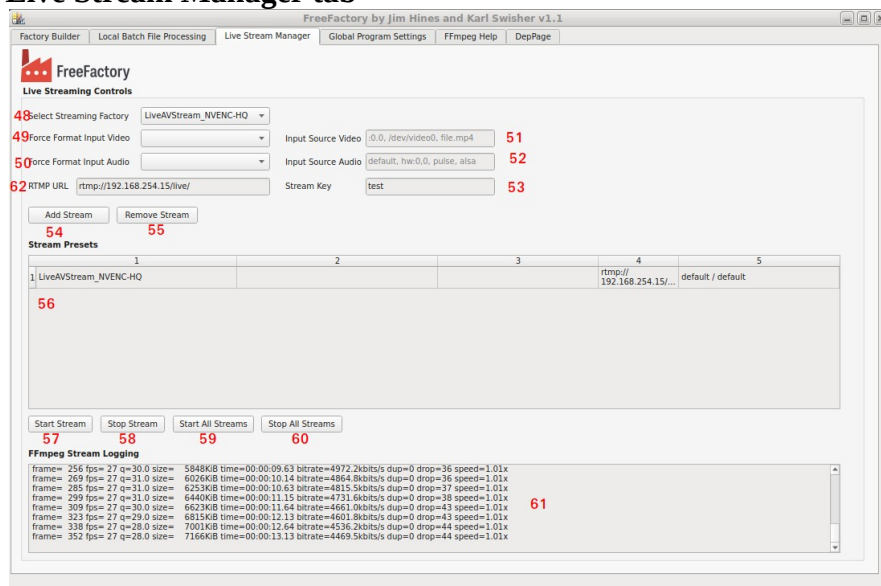
48. Select Streaming Factory: This works a bit different for now but will get an upgrade once a better way has been decided to accomplish this. You must first select your Stream Factory from the Factory Builder tab, then also select the same Factory here (yes, this is a bit cumbersome but it does work), then click (#53) Add Stream to add it to list. You must repeat this process for each Streaming Factory added.

49. **Force Format Video:** This can be any of file, x11grab, scbgrab or v4l2
50. **Force Format Audio:** This can be any of pulse, alsa or dshow
51. **Input Source Video:** This is a screen (:0.0), a video device or a file.
52. **Input Source Audio:** This can be default, hw:n,n, pulse, alsa, etc.
53. **Stream Key:** Unique stream key for your account.
54. **Add Stream:** Adds a stream to the Stream Presets list.
55. **Remove Stream:** Removes a stream to the Stream Presets list.
56. **Stream Presets:** This is a list of your streamable factories.
57. **Start Stream:** Starts the Selected stream.
58. **Stop Stream:** Stops the Selected stream.
59. **Start All Streams:** Starts ALL streams.
60. **Stop All Streams:** Stops ALL streams.
61. **FFmpeg Stream Logging:** Shows the output of FFmpeg during encoding/streaming.
62. **RTMP URL:** RTMP URL.

Global Settings tab

62. **Company Name:** (Optional) Enter your company name here.
63. Include in Metadata: This will include your company name in all metadata of converted files (Not yet enabled).
64. **Apple Delay:** This is deprecated and left in for backwards compatibility.
65. **Enable Factory:** This enables the factory (default). This is only used by the FreeFactory service.
66. **Remove Source:** Removes the source file after encoding. This is only used by the FreeFactory service.
67. **Remove Logs:** Removes the FreeFactoryConversion logs once completed. This is only used by the FreeFactory service.
68. **Default Factory:** Chooses a default Factory that is loaded whenever FreeFactoryQT is started (optional).
69. **Factories Path:** This allows users to move the /opt/FreeFactory/Factories folder to a another place or use multiple folders.
70. **FFmpeg Path:** Defaults to /usr/bin but can be changed to ie. /usr/local/bin.
71. **FFMx Program:** For future use.
72. **Free Factory Notify Folders:** Allows multiple watched folders. Useful for network shares. For future use.
73. **Free Factory Service Status:** This shows the status of the FreeFactory-Notify service. You must click (#76) Update Status to populate. All the control buttons below will also populate this window.
74. **Start (service):** FreeFactory-Notify service control.
75. **Restart: (service):** FreeFactory-Notify service control.

76. Update Status: FreeFactory-Notify service control.
77. Stop (service): FreeFactory-Notify service control.
78. Kill (service): FreeFactory-Notify service control.
79. Clear: Clears the Status Window.

FFmpeg Help tab
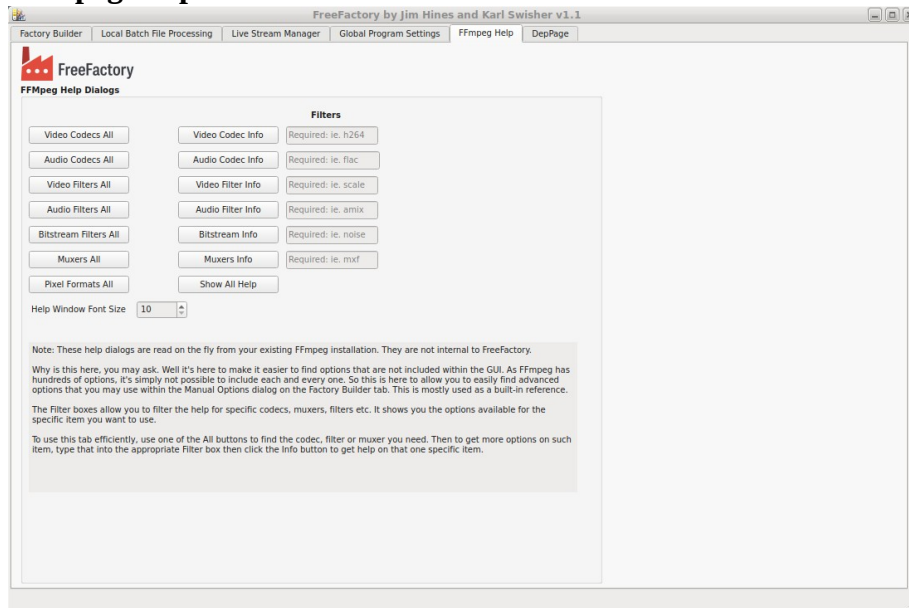
This tab gives you full access to the FFmpeg built-in help with filtering. It literally pulls this from FFmpeg itself so you will absolutely know what YOUR installed version provides.

DepPage tab

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

FreeFactory Service

Setting up FreeFactory service:

This requires TCL be installed on your system until this gets ported to Python3.

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 sudo)
3) Uninstall from USER mode
4) Uninstall from SYSTEM-WIDE mode
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.

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

The FreeFactory service is mostly useful for professionals 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. Each folder can have only ONE Factory assigned to it. 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 frontend 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.tcl 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. Once FreeFactoryConversion gets ported to Python3, we will set a job limit in Global Settings for this.