Automating FTP audio sources into Nexgen

using WinSCP scripting, AMB-OS and RCS Audio Format Converter

Contents:

1.) Overview of the FTP automation process

2.) Creating a Workable File Folder structure

3.) Using WinSCP to generate a script (batch file)

4.) Other means of getting audio, including regular AMB-OS and Windows powershell

5.) Using Windows Task Scheduler to run batch files

6.) Configuring Nexgen Auto Load

7.) RCS Audio Format Converter

 A.) Changing the default number of groups from 8 to your meet your needs

 B.) Creating DAT Templates for converted audio

 C.) Settings for audio conversion

 D.) Using Windows Task Scheduler to restart Audio Format Converter

**Overview of the FTP Automation Process**

In general, getting audio from individual client FTP servers via the internet is not a particularly difficult thing to do. Most clients provide stations with the server addresses, usernames and passwords and it is trivial to use free software such as FileZilla or WinSCP to set up a site in order to retrieve the audio from their servers.

Manually getting into the servers, converting the audio and then injecting it into Nexgen often requires a minimum of 8 to 10 steps from start to finish of the process. Moving the audio through the system manually not only is a time-consuming process, but also greatly increases the risk that a mistake might be made that may be a cause for loss of revenue or may simply be missed among the hundreds of pieces of audio that have to be handled in this way.

As an overview before we dive into the specifics of FTP automation, we first will create a batch file that will automatically log into the clients server, find the necessary audio, and download it into a file folder on our local computer. This also includes audio that gets downloaded through AMB-OS and is already sitting on the computer ready to be converted.

From there, we’ll use the RCS Audio Format converter software to convert the audio (generally in MP3 format) into a format that Nexgen can use. The format converter software also has the ability to create DAT templates that will inject each piece of audio with the necessary information it needs. Data like the cart number with cut, the name placed on the cart, expiration, archival and deletion dates if necessary and even the production category the audio will be placed into within Nexgen. At the end of the conversion, not only is the audio in its proper format, but it has also created a DAT file that Nexgen is ready to use.

Along with audio conversion and DAT template, the Nexgen “Auto Load” function needs to be enabled and running on the machine that the AFC software is running on. This function scans a particular created folder (say, a folder named “Converted Audio) constantly and if audio is present in the folder, Nexgen will load the audio into Nexgen with all of the information that was inserted by the DAT file and the audio is then able to be used by Nexgen.

**Using a Workable File Structure**

It goes without saying that the automation we are setting up can do everything that we need it to without creating any kind of file folder structure at all. Obviously, the audio can be downloaded into any generic folder, the batch files can be in a different folder and besides the one “Converted Audio” folder mentioned earlier, the process would move forward and work.

Yet, while the system will be totally automated, it occasionally will break. We are after all, talking about a computer system. AFC does hang occasionally (I have a fix for this later in the document), Windows will lose its mind, and Nexgen needs a restart at times. So, with that in mind, it is important to have a file folder system in place that will help you quickly determine where the audio for each show/client may be in the rare event that you have to manually enter audio once again.

A good starting place is to have a folder for each individual client (i.e. like “Focus on the Family”) and then to house all of that clients audio in that folder including the batch files that have been created to run for that client. You can use spaces, but often it may be better to take a name like “Focus on the Family” and simply call the folder “Focus” or “FocusontheFamily” as sometimes spaces generate a problem for the automated process. Preferably, you might also want to create sub-folders for “Daily”, “Weekly”, “promos” or any other special audio features that the client might also have in their production libraries. It might look something like the image 1 below:



**Using WinSCP to generate a Script (batch file)**

As mentioned earlier, the beginning step is to create batch files (executable text files) that upon execution will force a program to run without help from a user. It its simplest form, the batch file will use a program like WinSCP to log into an FTP server, use the GET command to retrieve the intended file, and then place it in a folder of our choosing. While there are several free or open-source FTP clients that can be used like FileZilla or WinSCP, WinSCP has the advantage of helping us to generate batch files that can be used to easily go and log into a machine. The nice part of it is that if you have already been using FileZilla to get into your FTP sites manually, WinSCP will import those login credentials you’ve already been using automatically so that you can continue to log into your FTP sites maually, but will also use those same credentials to generate the simple batch files we need for the next part of the process.

Once WinSCP is opened and you have logged into an FTP site, you’ll see a “Tabs” (Image 2 below) option at the top of the screen, and once opened, you’ll see an option called “Generate Session URL/Code”. Choose that option.

After choosing that option, you’ll be presented with a popup window that looks similar to image 3 below:

As you can see, there are 3 tabs for URL, Scripts and .Net assembly code. We want to chose the “Scripts” tab and then for format we want to use “Batch File”. This simple batch file is our jumping off point. The next step will be to hit the button labeled “Copy to Clipboard”.

From here, we want to open up the notepad app in Windows and paste the contents of the clipboard to the file. It is a good option at this point to begin saving the contents upon any change to the file that is made. When you save the first time, you want to name the file something sensible for the audio that you are getting (i.e., “FocusDaily”) but instead of saving the file as a .txt file, in the box under the naming box, you want to choose \*.\* (all files) and then append the extension “.bat” to the name of the file. Congratulations, you’ve just created a batch file.

Let’s look at the batch file WinSCP created (Image 4):

Above, we see what the batch file is trying to accomplish:

1. It gives the location of the program we want to use (WinSCP)

2. Tells WinSCP to open the account (username, password and server blurred out above) with associated keys

3. Offers commands (which we’ll get into momentarily) that let WinSCP know where the file we want is and also where to put the file on our computer which is hopefully the file structure we created earlier.

4. Exits the WinSCP program.

So, its a fairly simple operation so far, but the money step in the process is step 3 above. This is where much consternation will occur. It does so because FTP servers (mostly Linux operating systems) REALLY care about uppercase and lowercase, placement of spaces and the actual format of the file that you’re trying to get. In particular, most of the religious programming and others are not generically named by nature, they are meant to play on a particular date and are often named by date as shown below in Image 5:

As you can see, this is one of the “Commands” shown in Image 4, the GET statement is telling WinSCP “Since your logged in, go to the REMOTE folder ‘/Real Life Radio/Daily’ and then get the file with the following particular format “RLRX\_RLR5\_” that includes tomorrow’s date stamp in the format mm-dd-yy.mp2”. The directory location at the end of that line is the location on your LOCAL hard drive.

As you can see, it becomes imperative to not only pay close attention to the format and spacing of the file’s location on the REMOTE server, but also to the format and spacing of the LOCAL location you want the file to reside in. Fortunately, even though Image 4 shows two command options, generally the only command line that is needed is something like what you see in Image 5.

That command is about as difficult as it gets and once you have talked to your local person and figured out a particular client’s download schedule and whether they put shows in daily or all of the monthly shows in a single day, you can begin to suss that out for other clients as well in a similar command that generally looks like the one above.

At the end of this section, we have setup a single batch file to go and get audio off of a remote server and deposit it onto our local machine. This is repeatable for any number of clients and the thinking involved in the “Command” step is similar for each client. WinSCP helps setup the logging in portion, you have to setup the command to use. Fortunately, I’ve done a number of the common ministry client batch files already so in most cases it is a simple matter of changing your login credentials and the location you’d like the file to reside in on your local computer. That folder structure and batch files are available upon to request to CBC markets.

**Using AMB-OS and Powershell for getting audio from remote servers**

Using WinSCP is but one solution for getting necessary audio from remote servers. Most CBC markets also have AMB-OS receivers that pull audio from a satellite or straight from the internet. Essentially, Ambassador is transferring client programs that you have been given permissions to get from their individual folders on their FTP site. So clients will upload their audio to the Ambassador FTP site and if you have the appropriate permissions from the vendor, you will get them automatically sent via satellite or directly over FTP to you local machine. All of their audio files are labeled in shorthand similar to Real Life Radio in the example in image 5 previously.

Without going too far in the weeds regarding the AMB-OS UI that resides on the local machine that their satellite receiver pushes the downloaded files to on your LAN, you can do several things that will help in the automation process like renaming the files from AMB-OS short hand to something useful as shown in Image 6:

 Not only can you rename the files as shown above, but you also see that when the files are downloaded to your local machine, they can also be placed in the correct folders.

You may not care to use the renaming feature that the AMB-OS UI gives you, but it is helpful to place audio into a folder that separates it from the rest of the audio that AMB-OS places on your local machine.

Another way to handle this is via another simple batch file as shown below in the following example:

Similar to creating the batch file from earlier, this batch file simply moves the file from one folder to another. This can be done in a variety of ways, but the batch file simply uses the “move” command with no other commands in place. Again, it is important to pay attention to detail here and make sure that the everything is placed and spaced correctly.

Regarding Windows Powershell, this has only needed to be broken out a couple of times recently because the client provides a website to get audio from instead of an FTP site. Clients might prefer Dropbox or their own website to simply store documents and give links to them. WinSCP does not handle https:// at all, so another solution was necessary. Powershell is built into Windows and has an Integrated Scripting Environment (ISE) that you can use. Here is an example that follows and it is not for the faint of heart as the 5 lines shown took me a while to generate, but they do work and place the https:// audio file into the correct folder in my local drive. Obviously, I’d be able to help with something like this, but if you have a gift for writing scripts using these may not take you as long to crank out as it did for me. Similar to the WinSCP scripts, once you have worked through one that actually works, the thinking is nearly the same and you can make minimal changes and create another without much effort for a different client. One is shown below in image 8 as an example:

****

Breaking down the script, there is a variable called $Date that adds a day and formats it into a time related string. The $Url variable tells the web request where to go on the internet and the $Path variable tells the script where to place the audio on our local machine once it has gotten it from the web. Notice that both variables call the $Date value from the first line into use in order to get the correct audio. Lastly, the Invoke-Webrequest uses lines 2 and 3 to force the action of getting the audio file. It is IMPORTANT to note that Powershell scripts are not typically enabled to run on a install of Windows 10 and on by design as a security measure. In order to run a powershell script you’ll need to open the powershell app as an administrator and type in the following command “set-executionpolicy remotesigned” and hit enter. Then, the computer will run Powershell scripts that were created on the local machine.

Once we’ve proven that the audio has been gotten successfully using the batch scripts we’ve created with WinSCP, AMB-OS or Windows Powershell. It is now time to take the next step in the automation process and that is running these automation scripts at a given time.

**Windows Task Scheduler**

This middle task in the automation process is handled by a “behind the scenes” windows process called the Windows Task Scheduler. The same program that is used by Windows Update, your anti-virus software and a host of other windows processes to automate their activities can be used to automate the batch files that you have created in the previous 2 sections to download the audio that you need without your intervention. Searching for the “Task Scheduler” in Windows will open the app up on your window and on the right side of the opened window, you’ll see an option to “Create a Basic Task”. Making that choice opens up a window similar to the following which guides you through the process of scheduling your batch file to run at a particular time.

****

 As you can see, the beginning simply asked you to name your task and then you hit the “next” button. The “trigger” is where you actually schedule the event to take place, do you need it done daily, weekly or monthly? What about the time that you want it to run? All of those things need to be filled out before you move to the next step, which is the “action step”. The action step will require you to browse to where your actual batch file resides. The action step is going to run whatever executable file you put in place here. It may be a WinSCP batch file that goes out to an FTP site and grabs the next days audio and places in a folder. Maybe its the simple “move” batch file or possibly the Powershell script. No matter, once you have scheduled this basic task, it will run it at the time(s) that you have prescribed it to run.

I generally ask it to open the “Advanced” tab before I finish the creation of the task schedule. In this way, I go over all of the options. Some of the changes that I normally apply are to run the task “with highest privileges” and also to choose Windows 10 at the bottom of the first window. On another of the options on another tab, I “uncheck” the box for “If its on battery power, don’t run the task”. As a general rule, I don’t want anything to stop that task from running at the appropriate time.

There is no limit to the number of tasks that you can have the Windows Task Scheduler accomplish for you automatically without your input. Once you’ve created the task, you can “right click” on it and choose “run” to be sure that the task will run to your expectation. If it doesn’t, you may need to go back in and tweak your settings until it does run as expected. This exercise really completes the second step of the automation process. Up to this point, we now have created a batch file with outside programs that will be executed by Windows Task Scheduler and place our audio in the locations that we have chosen ahead of time. Before the final step on our automation is detailed, we have some configuration that needs to take place in Nexgen regarding it’s “Autoload” feature.

**Configuring AutoLoad in Nexgen**

The Autoload feature in Nexgen does exactly as it states. It is meant to be used in conjunction with the RCS Audio Format Converter to take its output and “autoload” it into the system. In order to properly use Autoload, we need to create a folder somewhere on the primary drive. The naming of your folder is left to your discretion, as an example, the Birmingham market calls ours “Converted Audio” and it is located on the root of the “[C:\](../../../C%3A/)” drive ([C:\Converted](../../../C%3A/Converted) Audio\). It will make sense shortly, but inside that folder are two other folders that we called “Final” and “Errors”.

You can find the Autoload in the “Utilities” menu of Nexgen. Choosing the option pulls up a window that looks like the following:

Going from top left, the Autoload will see if audio already exist on the system, it it does-we ask it to replace the audio with the new audio. It will then update the database and also overwrite the existing file. It will search the “Converted Audio” folder every 60 seconds and it if there is any audio that has been added to the folder it will process it and add it to the system. The 3 folders we created and named above are necessary to fulfill the obligations for the directories in question and they all need to be physically different locations. We simply leave the audio in the Converted audio directory as it will be replaced the next time the batch file runs. Lastly, you’ll hit the “Start” button and the feature will begin the process of consistently searching for new audio in the Converted Audio folder until you hit the “Stop” button (once the start button is clicked, it changes to stop).

This Autoload feature persists through restarts of Nexgen and reboots of the entire machine. If you find that your audio loading has halted, this would be the first place to look. My experience is that this feature is rock solid and hardly ever needs attention.

With the Autoload feature configured, the last step in the entire automation process is to configure the RCS Audio Format Converter to convert audio to the Nexgen standard MPEG 1 layer 2 format. The software also places metadata into the audio so that through the autoload feature, it will be placed into the correct production category with the cart/cut that you have planned out ahead of time.

**The RCS Audio format Converter**

The final step of the automation is the part that contains a fair amount of detail, but you’ve already done most of the hard work and you can see the finish line. The Audio Format Converter (from now on AFC) in simple terms is divided into singular groups for each piece of audio with settings about where audio can be found in specific folders (the inbox), the conversion folder (the outbox, whose value should always the “Converted Audio” folder we created in the previous step), the conversion format (for our purposes MPEG 1 Layer 2, with your choice for the compression value), the choice of a “DAT template” and any choices regarding normalization, creating VID files, deleting original audio files and running in “auto” mode.

The first thing you’ll note is that the default number of individual groups (files) in the initial install of AFC is only 8 groups. This low number would hardly be useful since our goal at the end of this process is to handle many more files than that number. So we go into the configuration section of the initial screen and choose for the number of groups to be higher. When choosing the number of groups, be careful to not overshoot your actual needs. This number can always be adjusted to a higher number and restarting the program will change the number of groups to the new setting. The issue with putting a higher number of groups than necessary in AFC is that it starts at the first group and cycles all the way through to the last so you’ll be adding unnecessary time between the AFC returning back to the first group.

Once the appropriate number of groups have been established, the next step is to create “DAT Templates”. These templates have all of the information that AFC needs in order to give each piece of converted audio the correct information from its cart number, cut number, audio name, end dates, archive dates, deletion dates and even the production audio category that the audio needs to be placed in. Below are two screenshots of a bunch of saved templates we have created and then one of the individual templates that are used:

 The DAT templates tab is where you find the opportunity to create new templates, edit templates, copy and delete templates. The next image shows an individual template and all of the settings that can be placed on a template.

As you can see, you can add as much or as little information as necessary. For the most part, the automation process is going to get daily programs once per day, so you are moving to a different paradigm regarding using day parts and putting audio into rotations. After you have your templates built out, one of the final steps is to go back into the individual groups and begin putting in the settings that you need for each piece of audio you want to use. The last image is the group settings:

 Typically, I’ll rename the group to something useful while keeping the group number somewhere in the title (normally at the beginning). Its good for me to know when a particular group is coming up for conversion/processing. The next thing to do is to click the inbox and browse to which folder the audio was placed into. The outbox will ALWAYS BE the “Converted Audio” folder. Unless you have a reason to convert to a different audio format, the format should be MPEG 1 Layer 2 with compression to your system’s taste. The DAT template that you painstakingly created can now be chosen for the audio group. We choose the options as above for all of our groups, but you may prefer more or less. Critically, “process in auto mode” must be chosen as AFC runs in auto mode as it moves from group to group checking to see if any audio has been added to each of the folders. “Trim for silence” and “Normalization” settings can be tweaked to your liking in the “Audio Processing” tab found in the configuration section window shown earlier in this section.

Once you are fine with all of the settings you have chosen, you can hit the “OK” button and then configure another group. If you have added all of the groups that you intend to add for the time being, you can simply hit the “Auto” button in the upper left of the main program window and the software will begin checking the inbox folders of each of the groups and processing the data. This is really the completion of the automation process.

When AFC finds audio in a group’s folder, you’ll see a light blue bar move across the screen as it completes the conversion and adds DAT settings. It will move onto the next group while it silently passes the audio to the Autoload utility. With the Autoload utility set at a 60 second check of the Converted Audio folder, it takes a minute or two for the processed audio to show up in the production category to which it was assigned depending on where in the 60 second window the audio was placed in the folder.

Lastly,the final batch files that you may want to create are ones that kill AFC once a day and then restart it a minute later. This can be done overnight so it does not interfere with any automatically processed audio. The only reason I suggest this is that it happens occasionally that AFC will freeze up and we’ve also had people shut the program down accidentally. Despite the confidence I have tried to engender in this process, we are dealing with computers and humans so it doesn’t hurt to have AFC automatically shutdown and then restart itself a minute or so later.

The batch file in notepad that needs to be created to terminate AFC simply has this single line in it:

 taskkill /f /im PsiAfc.exe

Remember to save it as a batch file and to use the Windows Task scheduler for it to work some time in the early morning—daily.

To restart AFC, you don’t need a batch file. You simply need to go to the Windows Task Scheduler and Create a Basic task as shown earlier in the document. Name the task, then configure the trigger (schedule is daily, preferably 1 minute after it was terminated above), and the action is to run the PsiAFC.exe file (browse to it in the C:\Programs Files (x86)\RCS\AFC4\ folder) and then run through the same settings I mentioned previously about highest privileges and so forth.

I hope this helps to take a load off of what you are doing on a daily basis so that you can focus on other details in your job of entering programs. As I mentioned, I have zipped folder of a number of ministries that already have batch files included and will just need to be tweaked to your individual market needs. Good Luck!