**Getting Started with Sonospy**

***(Adapted by Bartolomé Soriano from the project README, Wiki, information gathered from the Sonospy-Development Google Group and personal use)***

Mark Henkelis started developing Sonos related programs in 2009, and over time his work has grown to create the potential for a wide range of Sonos add-on extension & replacement facilities.

The project this document refers to is a tested and packaged subset of Mark's work. Essentially this subset works through the existing Sonos controllers, removing a number of limitations caused by the lack of resources in Sonos hardware. The current release of this project provides solutions to the following long standing community enhancement requests:

* Removal of the Sonos 65,000 track capacity limitation.
* Support multi-value tags, for example a track may be stored under both "Rock" and "Live Music" genres, or tracks or albums resulting from artist collaborations can be listed in the artist index under each separate artist.
* Improved and flexible music browsing capabilities, for example albums can be sorted by Date Released, and/or by Date Ripped, rather than just be listed in alphabetical sequence.
* Databases can be filtered for recently added music to your overall library.
* Playlists can be promoted to virtual album status, so that they appear in the indices as normal albums
* The major works of classical composers can become visible and selectable in the Sonos indices
* The range of music file types playable by Sonos is extended for all users, for example high-res FLAC files can be transcoded on-the-fly into something which Sonos can play

Sonospy provides a facility to index your local music library, and it then mimics something which Sonos will accept as a source of local library music. The standard Sonos controllers (CR200, CR100, Sonos Desktop Controller, Android, iPhone & iPad controllers) run without change, but are extended by the richer index which is served up to them.

The project is entirely standalone. It does not update your music tracks, nor does it alter any Sonos supplied software. The native Sonos controls and indices remain available, but new and more powerful facilities are available alongside them in the Music Services section of the controller.

The project is programmed in Python, stores its database using SQLite, and implements a proxy server facility for WMP, as well as a SMAPI service.

***NOTE:*** *I have only used Sonospy as a SMAPI service, so the configuration information provided will only cover that scenario.*

It is a cross platform project, and has been tested running in Linux, OS X and Windows environments.

***NOTE****: I have only used Sonospy under Windows (first Windows 7, then Windows 8 and 8.1), and OS X (10.9.2), so all configuration information/instructions will be provided only for those families of OS.*

*I am a novice user of OS X, so while I have tested successfully all of the configuration steps provided herein for OS X, there are most likely more efficient, and possible shorter ways to get the same results.*

It is built on top of the BRisa UPnP framework, with extensions added for Sonos ZonePlayers and the content they can serve/render.

**Packages bundled with Sonospy:**

The following packages are bundled with Sonospy (no need to download/install):

cherrypy: cherrypy WSGI server 3.1.2 (All platforms: <http://www.cherrypy.org/wiki/CherryPyDownload>)

python-brisa: python-brisa Framework 0.10.0 python-brisa Qt Reactor 0.10.0 (All platforms:<http://brisa.garage.maemo.org/installation.html>)

**Installation of Sonospy:**

1. Download Python 2.7, go to <http://www.python.org/download/releases/2.7.6/>,

* For Windows, select [Windows x86 MSI Installer (2.7.6)](http://www.python.org/ftp/python/2.7.6/python-2.7.6.msi) [(sig)](http://www.python.org/ftp/python/2.7.6/python-2.7.6.msi.asc) or [Windows X86-64 MSI Installer (2.7.6)](http://www.python.org/ftp/python/2.7.6/python-2.7.6.amd64.msi) [[1]](http://www.python.org/download/releases/2.7.6/#id5) [(sig)](http://www.python.org/ftp/python/2.7.6/python-2.7.6.amd64.msi.asc)
* For OS X, select [Python 2.7.6 Mac OS X 64-bit/32-bit x86-64/i386 Installer](https://www.python.org/ftp/python/2.7.6/python-2.7.6-macosx10.6.dmg)
* Install Python
* For Windows, ensure you add the Python install directory to your PATH environment variable.
  + For example, if you installed Python in the folder “C:\Python27”, the following command, run from a command window, will add that folder to the end of the existing PATH:

setx PATH “%PATH%;C:\Python27”

1. Download Sonospy from: <https://github.com/henkelis/sonospy/tree/msrefactor5>, use the button that says “Download ZIP” that appears on the right, near the bottom.

* Unzip to directory of choice
  + When you unzip, you will get a “sonospy-msrefactor5” folder underneath the directory you chose to unzip to. This “sonospy-msrefactor5” is the folder I refer to as the main folder where you unzipped the Sonospy installation later on in this document.

**Ports required for Sonospy to run**

The following ports need to be open in the firewall on the machine you are running Sonospy on. Most are standard UPnP/WMP ports, with additional ones for the BRisa framework and the proxy and SMAPI service.

Note that you may need to open ports specific to other UPnP media servers too (if you are using them).

The following ones are the ones I have opened through the firewall, and are working well:

2149 UDP (current M-SEARCH send port so WMP is picked up)

10280-10284 UDP

50111-50119 TCP (control point, proxy and service, can be changed in the ini)

For Windows:

Create both incoming and outgoing rules to open these ports through the Windows Firewall.

If the WMP proxy (not covered in this document) would be used, port 10243 would have to be open also. It is important to note that this would conflict with Windows Media Player if you are running it in the machine too.

For OS X:

It is important to check that the OS X Firewall is correctly configured to work with Sonos. Please look at the following page from Sonos support on how to accomplish this:

<https://sonos.custhelp.com/app/answers/detail/a_id/2069>

The easiest way I have found to open the ports in OS X Mavericks, under the assumption that the Firewall is turned on in your Mac, is through the following procedure, from a Terminal window:

sudo vim /etc/pf.conf

Add the following lines to the config file (after the com.apple anchor is fine) where en0 is your Ethernet adapter and the ports are those listed above:

pass on en0 proto TCP from any port 50111 to any port 50111

pass on en0 proto TCP from any port 50112 to any port 50112

pass on en0 proto TCP from any port 50113 to any port 50113

pass on en0 proto TCP from any port 50114 to any port 50114

pass on en0 proto TCP from any port 50115 to any port 50115

pass on en0 proto TCP from any port 50116 to any port 50116

pass on en0 proto TCP from any port 50117 to any port 50117

pass on en0 proto TCP from any port 50118 to any port 50118

pass on en0 proto TCP from any port 50119 to any port 50119

pass on en0 proto UDP from any to any port 2149

pass on en0 proto UDP from any to any port 10280:10284

The first line allows TCP data on port 50111 to pass in and out of your machine via en0. This allows it from any IP address, but you could have changed the first mention of “any” to an IP address or IP address range, in order to only allow from a single IP address or IP address range. The subsequent lines perform similar functions for the other ports.

For more information on the pf.conf file, please see the man pages for it, or visit:

https://developer.apple.com/library/mac/documentation/Darwin/Reference/ManPages/man5/pf.conf.5.html

If you aren’t using a wired connection, you can change en0 to en1, or if you’re uncertain which adapter you’re using, run the following command to see which is active on your setup, noting which has a status of “active”:

ifconfig

This should yield something like the following. Note how en0 is “active”.

en0: flags=[redacted]

options=[redacted]

ether [redacted]

inet6 [redacted]

inet [redacted]

nd6 options=[redacted]

media: [redacted]

status: active

en1: flags=[redacted]

ether [redacted]

nd6 options=[redacted]

media: [redacted]

status: inactive

Save the config file and run the following to reload the config file, and verify using verbose:

sudo pfctl -vnf pf.conf

sudo pfctl -Rf pf.conf

***Note: Due to behavior by the Sonos Mac Desktop Controller and the way it shares ports, it is important that you start the Sonospy server BEFORE you run the Sonos controller. If not, you will not be able to play any tracks from Sonospy.***

**Sonos Settings:**

Since for the purposes of this document, we will only be using the SMAPI facility of Sonospy, no additional settings are required in your Sonos system.

**Using Sonospy:**

**Overview**

* 1. The first step is to create the Sonospy index over your local library music shares.
  2. The next step is to configure the pycpoint.ini file before the first run of the Sonospy server
  3. The next step is to start the Sonospy SMAPI service and connect to your Sonos system
  4. You can immediately use the new facilities via your Sonos controllers
  5. Periodically you run an update scan, causing the Sonospy index to pick up newly ripped items, new music downloads, and to see any tag changes or other changes that you have made in your music.

**Step 1: Create the Sonospy index of your local library:**

Sonospy is highly configurable, to suit a wide variety of user preferences.

Indexing your whole library is just one command, but it may take a while depending upon the size of your library and the speed of your computer. The Sonospy index design has traded off index size for processing speed. As a rule of thumb each track will take about 5k to 6k of disk storage for the index.

For the above two reasons it is recommended that you initially just index a manageable subset of your local library to get an initial taste of what Sonospy can offer. This way, before committing to indexing your whole collection, you can make sure that you have Sonospy configured the way that you want it, and you can also verify that your installed copy of Sonospy is working correctly.

Identify a path, or a couple of paths, containing a subset of your local music library. Ideally the directory tree beneath the chosen path(s) should contain at least a couple of albums. If your music library isn't segmented into separate paths, you could copy some tracks to a path somewhere else on your disk. Scan will work recursively to scan any subfolders from the origination of the scan.

Here are some example commands, for Windows and OS X, to create the initial database. In these cases, the name of the database created is Sonospy.db, and will be placed in the “sonospy” folder underneath the main folder where you unzipped the Sonospy installation.

*These scan commands are run from the “sonospy” folder underneath the main folder where you unzipped the Sonospy installation.*

*Run the commands from a Command Window (cmd.exe) in Windows, or from a Terminal window in OS X.*

In the first Windows example, the music folders are in a NAS, and the root path to access is [\\BaLiDS\music\BaLi Music Collection\Music](file:///\\BaLiDS\music\BaLi%20Music%20Collection\Music).

In the second Windows example, the music folders reside in the path X:\BaLi Music Collection\Music, and drive X: could be an external drive connected to the machine running Sonospy, or could be a NAS folder mapped as a network drive.

Note that in Windows every backslash (\) in the path needs to be escaped with another backslash (\). Also, there needs to be a backslash (\) before each space in the folder name, or you need to put the whole path name within single quotes (‘).

In the OS X example, the music tracks are located in, or beneath, "/mnt/nas/music".

Note that in OS X, if the path contains spaces, you will have to place two backslashes (\) before every space, or put the whole path name within quotes (“), and one backslash (\) before every space.

For windows with a NAS:

python scan.py -v -d Sonospy.db \\\\BaLiDS\\music\\BaLi\ Music\ Collection\\Music\\Aerosmith > Test.log

Or windows without a NAS

python scan.py -v -d Sonospy.db X:\\BaLi\ Music\ Collection\\Music\\Aerosmith > Test.log

OS X:

./scan.py -v -d Sonospy.db /mnt/nas/music/38\\ Special > Test.log

Another thing to note is that the -v option places Sonospy into verbose logging mode. The Test.log file may prove useful if there is a set up problem. You shouldn't use the -v setting when you scan your whole library as the log will be too large to be of any use.

When you do a scan, logs are written to the “sonospy/logs” folder, including log/warnings and errors - there is a separate one if you specify -v (scanlogverbose.txt). The verbose log in that folder can provide much more information to aid in troubleshooting any scan problems.

If you want to extend your test database with further music, you can run the scan command multiple times, each time to the same Sonospy.db database, but pulling in music from a different music location (i.e. add music from another artist, to the music from Aerosmith or .38 Special already indexed by the above examples).

You have now created the Sonospy index over some of your music. You are ready to move on to step two, where you will use that index via the Sonos controllers.

When you have tested your installation of Sonospy, and have tailored it to your satisfaction, you need to return to this step to index you whole local library. You use the same command, this time without the verbose logging option, and this time running over your whole library not just a few albums. You can use the same Sonospy.db database that you created above, unless you included music into the test, which you don't want to see in your main database. In that case you can delete the test copy of Sonospy.db before proceeding.

Example commands follow, although obviously you need to tailor them for your own music location(s). Again, these commands are run from the “sonospy” folder underneath the main folder where you unzipped the Sonospy installation.

For windows with a NAS:

python scan.py -d Sonospy.db \\\\BaLiDS\\music\\BaLi\ Music\ Collection\\Music >SonospyScan.log

Or windows without a NAS

python scan.py -d Sonospy.db X:\\BaLi\ Music\ Collection\\Music >SonospyScan.log

For OS X:

./scan.py -d Sonospy.db /mnt/nas/music > SonospyLoad.log

If you have more than one music share, you can index them one at a time, or all at once. On the first run, if scan sees that the named database does not already exist, it will automatically create it. On any subsequent runs, when outputting to a pre-existing database, it will just add in the new material. In fact, anytime you add new music to your collection, you will want to re-run your scan to make sure it gets added to the Sonospy database.

The command format to handle multiple music location paths, during a single pass, is to list them one after the other, with a blank separating the paths. Here is an example:

For Windows:

python scan.py -d Sonospy.db c:\\music c:\\OtherMusic >SonospyLoad.log

For OS X:

./scan.py –d Sonospy.db /mnt/music /mnt/OtherMusic >SonospyLoad.log

You will find in the “sonospy” folder under the main folder where you unzipped the Sonospy installation, a “scan.ini” that controls the behavior of the scanning process. This file is very well documented inside, so please take a look at each of the options within it and change them according to your preferences.

In order to make it easier to run the scan command, it is recommended to create a .cmd (Windows) or .sh (OS X) file to run the scan, and place it in the main Sonospy folder. For example, you could create a “SonospyScan.cmd” or “SonospyScan.sh” file containing the following lines:

For Windows:

cd sonospy

python scan.py -d Sonospy.db \\\\BaLiDS\\music\\BaLi\ Music\ Collection\\Music >SonospyScan.log

If you do not want the command window to remain open when this batch file is run, you can replace the “python” command with “pythonw”.

For OS X:

#!/usr/bin/env sh

# -\*- coding: utf-8 -\*-

cd sonospy

./scan.py -d Sonospy.db /mnt/nas/music > SonospyLoad.log

Remember to make this file executable.

Of course, you would replace the name of the database, music collection path and log file with your preferred values.

Thus, every time you need to update your Sonospy index, you would run “SonospyScan.cmd” (Windows) or “SonospyScan.sh” (OS X).

**Step 2: Configure the pycpoint.ini before first run of Sonospy server**

The pycpoint.ini, located in the “sonospy” folder under the main folder where you unzipped the Sonospy installation, controls many of the features of the server.

Open this file with your preferred editor. The critical section to modify is the [INI] section. This is exactly what I have placed in mine:

controlpoint\_port=50111

proxy\_port=50112

comms\_port=50113

smapi\_port=50114

wmp\_proxy\_port=50118

wmp\_internal\_port=50119

internal\_proxy\_udn=uuid:5e0fc086-1c37-4648-805c-ec2aba2b0b27

You can replace what is in that section with the above lines.

This uses the ports we opened in the firewall and provides a unique UDN for the Proxy.

The other sections of the .ini are very well-documented, and very easy to change. Please look at them and change any of them according to your preferences.

Once you are done with the changes, save the file.

**Step 3: Start the Sonospy proxy server:**

The Sonos controllers allow you to browse your local library music by either their own restricted index, via the WMP server from Microsoft, or via SMAPI services. One of the methods Mark has implemented is a SMAPI service in Sonospy, and through this method he is able to offer his extensions to Sonos users, including search.

This step starts up the Sonospy server function, which will use the Sonospy database created above, and translate this information into something which Sonos is expecting and can understand.

The native Sonos capabilities will remain available, and unaffected, in the Music Library section of the controllers. The new Sonospy indices are also available, alongside, in the Music services section in the Sonos controllers.

Here are example commands to start the Sonospy server. For Windows, the command is run from the “sonospy” folder underneath the main folder where you unzipped the Sonospy installation. For OS X, we can use the “sonospy\_proxy” shell script provided in the main folder.

*As with the scan commands, these commands are run from a Command Window (cmd.exe) in Windows, or from a Terminal window in OS X.*

For Windows:

python pycpoint.py -p -sSonospy=Sonospy,Sonospy.db -r

For OS X:

./sonospy\_proxy -sSonospy=Sonospy,Sonospy.db -r

This will start the Sonospy server as a SMAPI service named “Sonospy”, using the database named “Sonospy.db”, and will register the service.

Similar to the scan process, it is a good idea to create a .cmd (Windows) or .sh (OS X) file to start the server, and place it in the main Sonospy installation folder. For example, you could create a “SonospyStart.cmd” or “SonospyStart.sh” file with the following content:

For Windows:

cd sonospy

python pycpoint.py -p -sSonospy=Sonospy,Sonospy.db -r >../pycpoint.log 2>&1

If you do not want the command window to remain open when this batch file is run, you can replace the “python” command with “pythonw”.

For OS X:

#!/usr/bin/env sh

# -\*- coding: utf-8 -\*-

./sonospy\_proxy -sSonospy=Sonospy,Sonospy.db –r

Remember to make the OS X .sh file executable.

This will start the Sonospy server as a SMAPI service named “Sonospy”, using the database “Sonospy.db”, registers the service, and output any errors and messages to a log named “pycpoint.log”, in the same folder as the .cmd file.

Of course, you would replace the service name and database name with your preferred choices.

**Step 4: Use the new facilities via your Sonos controllers:**

The Sonos controllers should automatically detect and list the Sonospy controller (identified with the title "Sonospy", under “Add Music Services”, if you used the example command shown above.

Sonos sees this as a new Music Service, so it will be found at:

* Desktop Controller (Windows, OS X): “Add Music Services” button, under “Select a Music Source”
* iOS and Android Controller: “Add Music Services” button

If you used the example commands provided in this document, you would find a music service named “Sonospy”, which you would select. Sonos will ask you to add the Sonospy service to your Sonos system, you would press “Next”, then “Done” in the next screen, and the service is added to your Sonos system.

When you open this Sonospy music service you should see the familiar indices (Artists, Albums, Composers, etc). What you see inside these indices is flexible, and can be easily tailored for your own preferences.

By default, Sonospy mimics the indices that Sonos shows. Please read through the [indexing] section of the pycpoint.ini file for an explanation on how to customize the indices and searches that are shown.

As explained there, in order to specify your own indices and searches, you can add the name of an ini file after the database name in the command to start the server. The examples below show that. Remember that for Windows, the command is run from the “sonospy” folder underneath the main folder where you unzipped the Sonospy installation, and for OS X it is run from the main folder.

For Windows:

python pycpoint.py -p -sSonospy=Sonospy,Sonospy.db,userindex2.ini -r >../pycpoint.log 2>&1

For OS X:

./sonospy\_proxy -sSonospy=Sonospy,Sonospy.db,userindex2.ini -r

In this example, the “userindex2.ini” file will be used to determine the indices and searches that will be shown in the Sonos controllers for the Sonospy music service.

I recommend reading through the “defaultindex.ini”, “userindex.ini” and “userindex2.ini” files, located in the “sonospy” folder under the main folder where you unzipped the Sonospy installation, to get a thorough idea of the many capabilities of Sonospy in this regard.

**Step 5: Running an update scan to detect new and changed music:**

Run the .cmd (Windows) or .sh (OS X) file you created for the scan process (if you followed the example provided before, this would be “SonospyScan.cmd” or “SonospyScan.sh”). This will update the Sonospy database, adding any new music and updating the tags of any changed music in your collection.

**Stopping the Sonospy server**

For Windows:

To stop the Sonospy server, I usually just close the Command window which remains open while the server is running (I minimize it when I run the “SonospyStart.cmd”). If you used “pythonw” instead of “python”, the command window does not remain open.

Also, it is important to end the “python.exe”, or “pythonw.exe” (depending on which one you used) task via Task Manager, or else the server will not start when you try to restart it.

For OS X:

Use the “sonospy\_stop” shell script which is located in the main folder where you unzipped the Sonospy installation. Run this command from a Terminal window:

./sonospy\_stop

I stop the server before making any changes to any of the .ini files (pycpoint.ini, and in my case userindex2.ini), make the changes, and then restart the server.