# Creating an MSP driver suite to support grouping and independent service drivers

The aim of this document and the associated Lua and XML code snippets is to provide an example of how to modify or create an MSP driver for a music player Device to support placing individual service drivers under Listen and to support grouping one player with another. The code snippets are taken from the Sonos 8.0.3 driver suite and are intended as an example to build from rather than a complete, working solution.

The Suite comprises of snippets in four different driver packages, all of which are designed to work together. They are:

**Network** – This is the central connection point for all service, line-in and player drivers. It stores the information about each connected device and shares it to all other devices so that each driver is aware of the entire suite. The Network driver uses the AVSwitch proxy.

**Player** – This is the driver designed to manage a connection with a single player. It provides for transport control, metadata feedback and the connections for Director to know where the device is physically connected. It may or may not also allow direct browse capability depending on application. The Player driver uses an MSP proxy and an Amplifier proxy.

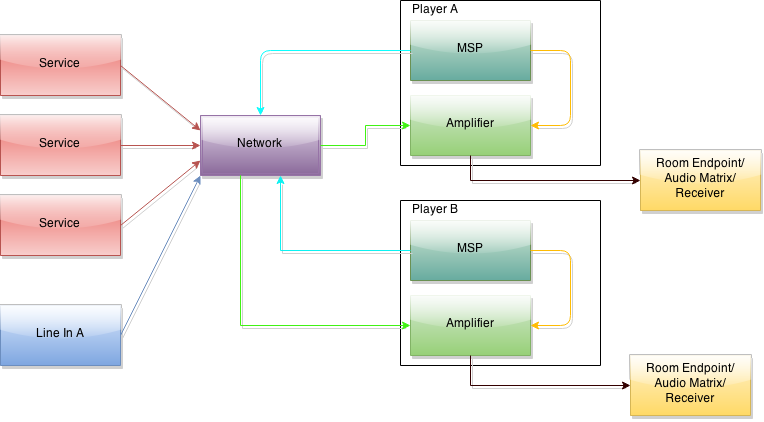
**Service** – This is the “minidriver” that creates a viewpoint directly onto each service offered by the player platform. It uses information about the Players provided by the Network driver to make a connection to the API for the specific service required, so that the end user has a direct ability to browse and search the selected service from the Listen page in Navigator. The Service driver uses the MSP proxy, and is likely to reuse a large amount of code from the Player driver if the Player driver was written to be able to browse and search all services/libraries supported by the Device.

**Line-In** – This optional driver is for systems where the line input on one player can be shared to all other players. It uses the AVSwitch proxy.

The general philosophy of the Suite is to allow the Players to be configured however you wish:

1. A single, centralized Player with line outputs into a matrix switch for home-run distribution
2. As in (1), but with multiple centralized Players to provide additional streams of audio
3. One or many single-room Players with speakers providing a single stream(s) of audio to a single room(s)
4. A mix of (3) with either (1) or (2)

## XML and connections between components of the Suite



This diagram shows the interconnections between the various components of the Suite. Each of the different colors of line represent a different range of binding IDs with common class in the XML.

* Blue indicates a line-in > network binding of class RF\_XXYYZZNET\_LINEIN
* Turquoise indicates a player (MSP) > network binding of class RF\_XXYYZZNET\_ZONE
* Red indicates a service > network binding of class RF\_XXYYZZNET\_SERVICE
* Green indicates a network > player (AMP) binding of class RF\_XXYYZZNET\_AUDIO
* Yellow indicates a player (MSP) > player (AMP) binding of class RF\_XXYYZZ\_ MUSIC
* Black indicates a player (AMP) > room endpoint, receiver, matrix, amplifier etc and is the “actual” output of the player device.

These classes should be renamed to remove the XXYYZZ and replaced with an appropriate, unique descriptor. In the Sonos driver Suite, this XXYYZZ is replaced with SONOS8.

The reason for using RF connections is that we can give them custom names like this. “Regular” connections like STEREO, DIGITAL\_OPTICAL etc cannot be extended and so provide the dealer the option to incorrectly make connections from parts of the Suite to other parts of the Suite or other devices entirely. By using these custom classes we can take away this possibility of incorrect bindings.

In addition, we use autobinding connections to ensure that the dealer doesn’t have to make even these connections. In the Sonos driver suite, these autobind IDs start with 252 and then 01 through 05 for the different classes; you should pick (and use for the entire suite) a random two or three digit number to replace the XXYOURNUMHEREXX in the XML so that you will end up with a four or five digit number that should be reasonable unique.

Not shown in the diagram are the “local” line inputs available on the Amplifier component of the Player driver. These local line inputs are only available in Control4 in rooms where the Player is on the direct path; they will not use the Network to route to other players.

The example driver has one connection which can be of class DIGITAL\_OPTICAL, DIGITAL\_COAX or STEREO to allow for the dealer to use a DAC or ADC of their choice and also to allow for variance between models. If your device has more than one line input AND different inputs are capable of being active at the same time, you will need to customize this section of the XML.

The Line in driver shown in the diagram is designated as a “global” line in, and allows for the input from any device to be selected on any other device through the Network. Again, if your device has more than one selectable line in AND each input can be selected discretely by any other player, regardless of how the device with the line ins is configured or what it is playing, you will need to add some configuration options to the Line In driver to select which of the device Line inputs you are making available.

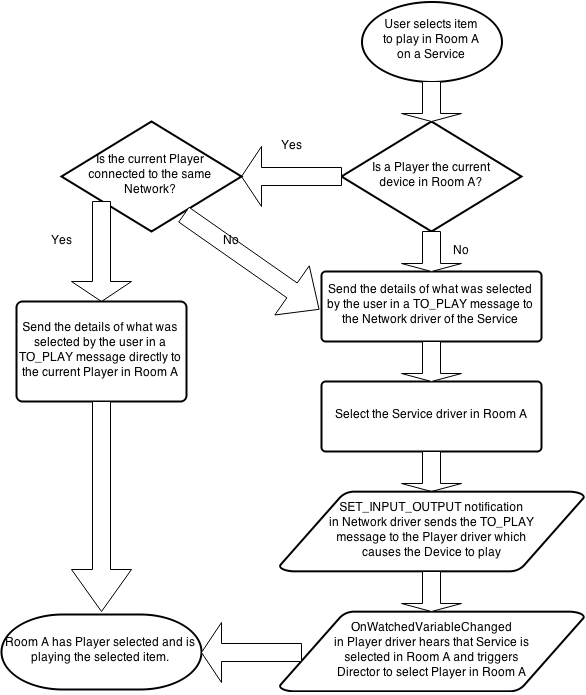
The exact configuration of the XML connections may need to be altered if:

* The Player hardware does not support the concept of “grouping”, whereby one player is designated as responsible for managing the stream of audio and other players are able to render the same stream at the same time, with transport control of the entire group managed by the “master” player (in this case, you would remove the Turquoise connections from the Suite)
* Line inputs (if any) are not able to be streamed from a device independently of what that device is currently playing (i.e. the Aux input on Player 1 can be selected by Player 2, regardless of whether Player 1 is already rendering that Aux input or indeed if Player 1 is rendering an entirely different stream) (in this case, you would remove the entire Line in driver and all relevant connections from the Suite)

The reason we also have the local connection from the Player MSP to the Player Amp is to give Director a shorter path to choose when selecting the Player MSP driver. This encourages Director to not use the Network in situations where it is not needed.

## Lua and using ReceivedFromProxy / OnWatchedVariable changed to trigger playback

When a user has drilled down into a Service driver to the point they are selecting an item to play, several things are accomplished by the various items in the Lua code. The following diagram shows the process:



This method will allow Director to select the appropriate source for music to play if a Player is not already selected in a room. If one is selected in a room, that Player will be re-used to play this audio. Allowing Director to select the best path allows the use of multiple Devices available to multiple rooms without needing the user to decide which of these Devices should be used first. If that functionality is required, the user will need to first select the Player corresponding to the Device in the room, then choose from the Service.

The supplied code is not a complete driver; the XML and Lua both need to be incorporated into an existing driver structure in order for them to work. This process may require some re-engineering of the driver Suite to provide complete functionality.

## Being a good developer

As mentioned in a number of places above, some customization of the provided XML is required to integrate it correctly; we want to make sure that drivers from different Suites won’t auto connect to each other so choosing good class names is important. We also already have some drivers in the system named “Rhapsody”, “Pandora” etc and will obviously get more as more MSP driver Suites become available. These drivers should have the manufacturer set to your Company Name (as you would with the Player driver) so that dealers can make sure they are selecting the correct driver. The <name> tag in the driver should also contain this information. For example, the Rhapsody driver as part of the Sonos suite has a tag of <name>Rhapsody for Sonos</name>. This is for the same reason.

Any questions on the supplied code snippets or this document should be sent to William Walsh [wwalsh@control4.com](mailto:wwalsh@control4.com) and if necessary, this document or additional snippets will be provided and this package updated.

This document and all code snippets in this package are Copyright 2015 Control4 Corporation. All rights reserved.