Technical Note: Controlling a Hyperdeck

Author: John R. Naylor
Date: July 25, 2014
Copyright © 2014, Ross Video Ltd. All Rights Reserved

1 Introduction

At NAB 2014 we demo'd a Blackmagic Designs' Hyperdeck being controlled from a DashBoard Custom Panel. An updated version of it accompanies this document which describes how it was done, and highlights some of the implementation specifics.

Gaining control over devices via Ethernet requires the programmer to understand the device's control protocol, and handle its responses and exceptions. Ross Video cannot provide end user support for integrations of this type, but will provide appropriate assistance to programmers engaged in creating them via tutorial videos, and publications like this one.

The integration documented here shows how DashBoard can load a list of clips from the deck, select, cue and play them, and adjust the play speed. It also allows the user to set up recording sources, and to crash record clips. Extending the panel to, say control multiple decks that can record as a gang would be useful. If you do this, please consider sharing it with the growing DashBoard Custom Panels community

2 Technical Details

The system diagram for this integration is straightforward.

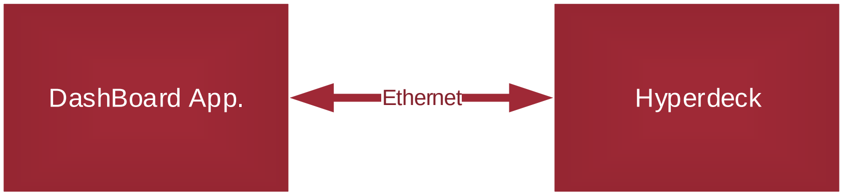


Figure 1 - System Diagram

The only piece of information needed is the deck's IP address which can be read or set using the deck's setup menu.

2.1 You will need…

Item	Supplier	Notes
DashBoard 6.2 Beta	Ross Video, Ltd	This should work in the 6.1 released version of DashBoard too.
A Hyperdeck	Blackmagic Designs	I've used both the Studio and Studio Pro successfully.
An SSD drive	Various	Recording media

Please follow the installation instructions for the packages listed above carefully.

2.2 You should know…

Item	Resources
Basic PanelBuilder Scripting	PanelBuilder 105 at DashBoard-U
Using Functions in PanelBuilder	PanelBuilder 203
Using Network Listeners in PanelBuilder	PanelBuilder 204
JavaScript	W3 Schools
Hyperdeck command protocol	Blackmagic Designs Hyperdeck Manual, pages 57-71

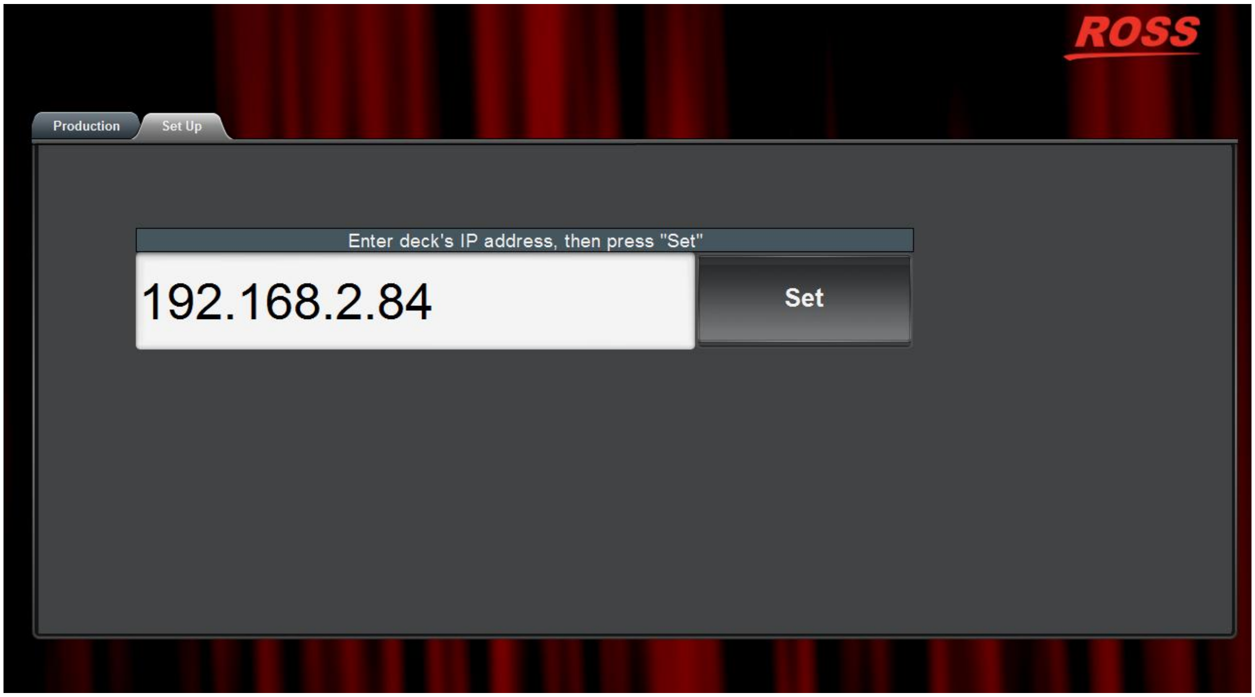
Note that the integration documented has worked on Windows 7.1 Pro 64 bit. It should work on other supported platforms but I've not done this and, regrettably, cannot offer support should you encounter difficulties on Mac OS X or Linux.

2.3 Using the Custom Panel

First ensure that the deck will accept remote control by pressing its <REM> button so that it is illuminated.

Now launch the demo Custom Panel.

Select the <Set Up> tab, enter the deck's IP address, and hit the <Set> button.



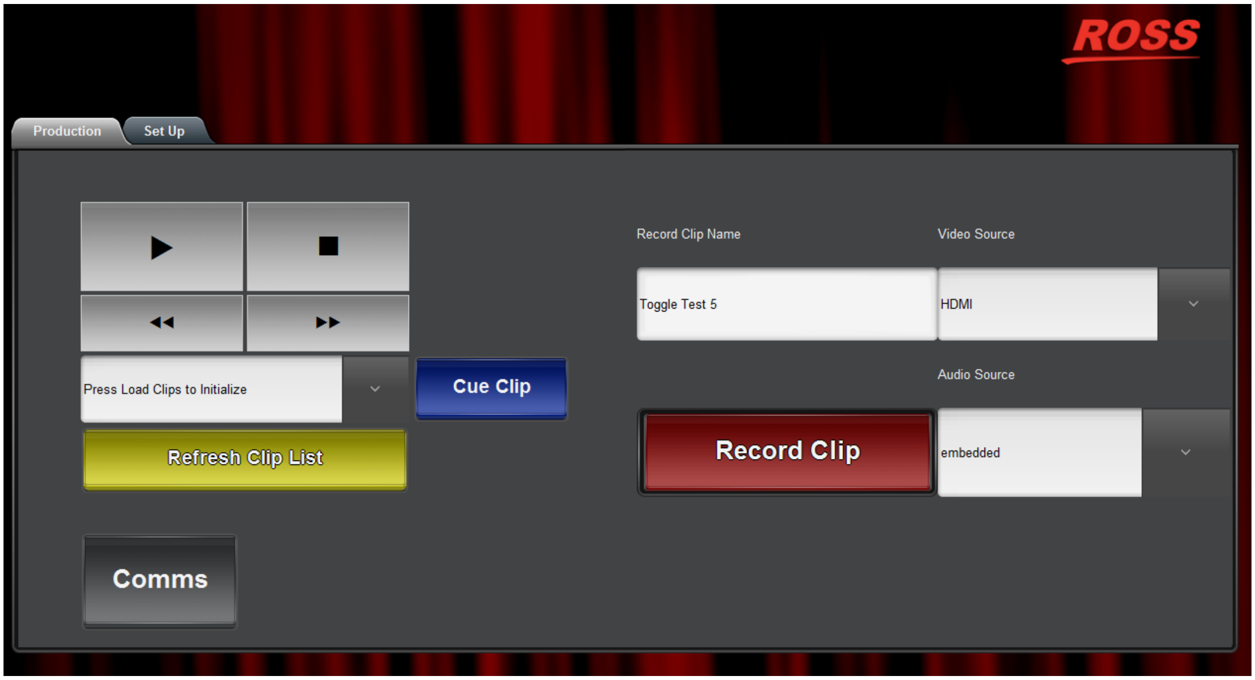
Enter deck's IP address, then press "Set"

192.168.2.84

Figure 2 - Set Up

Tip

If you want to avoid this step because you're always using the same deck at the same IP address, you can enter the IP address in the Custom Panel's hosts table. Do this by entering edit mode. Double click the background to bring up the editing GUI, and use the right hand tree navigator to select the hosts lookup where you'll be able to overwrite the Panel's default IP address.



Press Load Clips to Initialize

Record Clip Name: Toggle Test 5

Video Source: HDMI

Audio Source: embedded

Figure 3 – Production Panel

To play back previously recorded clips, press <Refresh Clip List>.

This should load the clip names in to the dropdown above this button, and the Comms button should blink momentarily. If this doesn't happen, there is some sort of problem. Open the debug view and try again to see whether any problems are reported there.

You can now select a clip, cue it using the <Cue Clip> button, and use the transport controls in this part of the panel to play & stop your clip and adjust its playback speed.

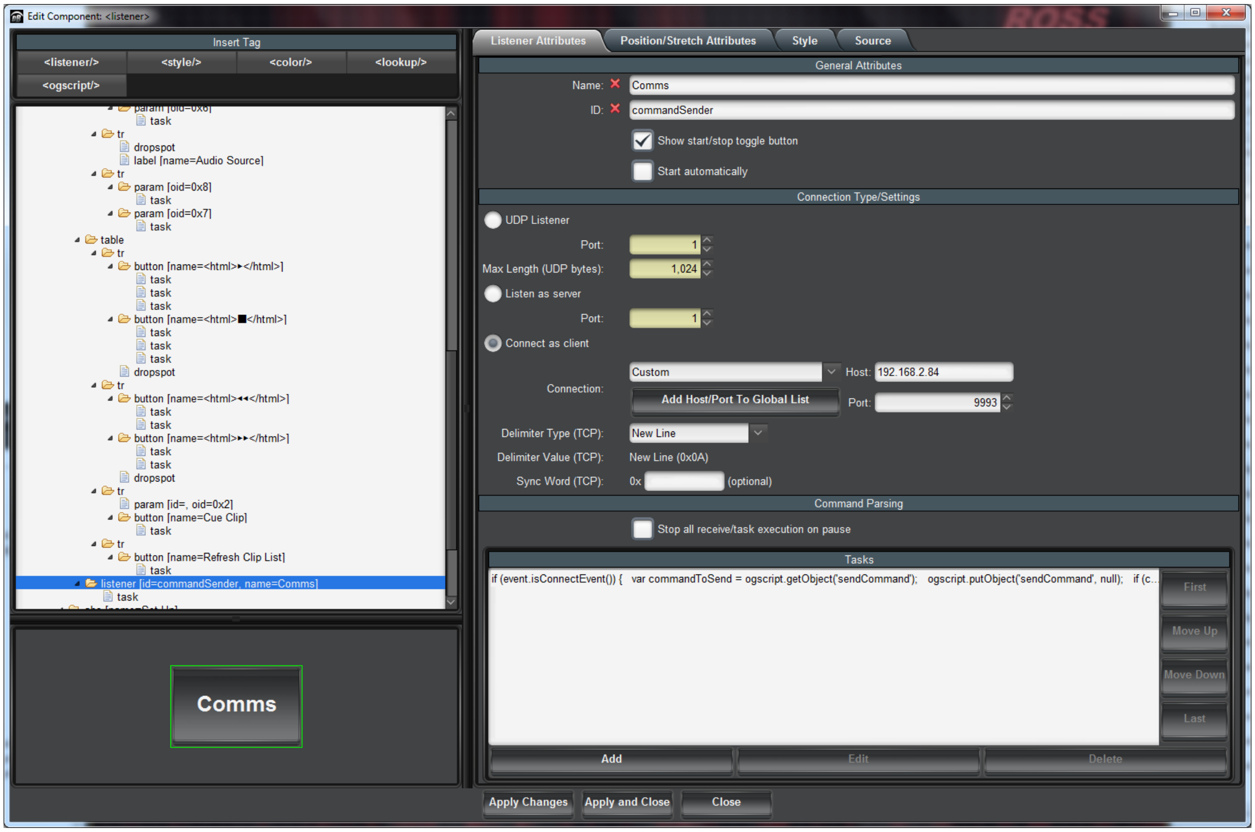
The recording controls provide you with the ability to set up the audio and video sources to record, and a name for the clip. Pressing the <Record Clip> button will start the recording.

2.4 A look at the code

The TCP listener is the key to this integration. This panel also demonstrates how to use global Javascript variables within a Custom Panel that is also a useful technique.

Figure 4 shows how to set up the Listener correctly for interoperation with the deck. The host information is loaded from one of the Custom Panel's internal parameters that is initialized from a default value when the panel is loaded. The hosts' table is illustrated in Figure 5. Edit this to change the default value of the deck's IP address.

Note that Hyperdeck communicates on port 9993 which is also stored in the hosts table.



```
      task
  tr
    dropspot
    label [name=Audio Source]
  tr
    param [oid=0x8]
      task
    param [oid=0x7]
      task
table
  tr
    button [name=<html>▸</html>]
      task
      task
      task
    button [name=<html>■</html>]
      task
      task
      task
    dropspot
  tr
    button [name=<html>◂◂</html>]
      task
      task
    button [name=<html>▸▸</html>]
      task
      task
    dropspot
  tr
    param [id=, oid=0x2]
    button [name=Cue Clip]
      task
  tr
    button [name=Refresh Clip List]
      task
listener [id=commandSender, name=Comms]
  task
```

Name: Comms

ID: commandSender

☑ Show start/stop toggle button

☐ Start automatically

○ UDP Listener

Port: 1

Max Length (UDP bytes): 1,024

○ Listen as server

Port: 1

● Connect as client

Connection: Custom Host: 192.168.2.84

Add Host/Port To Global List Port: 9993

Delimiter Type (TCP): New Line

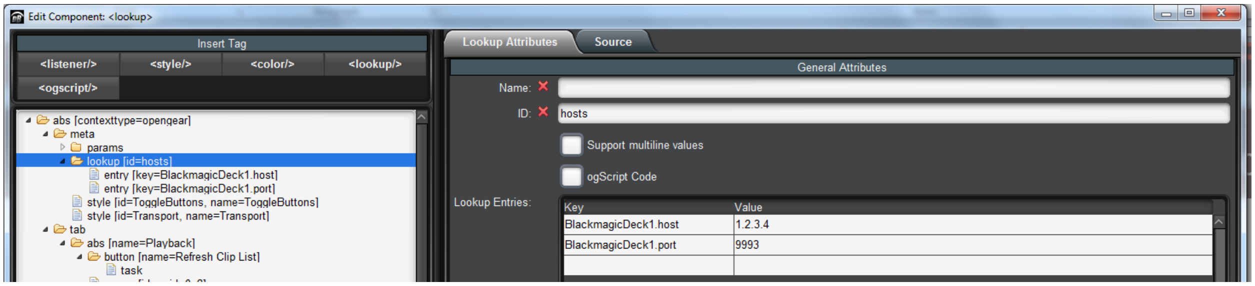
Delimiter Value (TCP): New Line (0x0A)

Sync Word (TCP): 0x (optional)

☐ Stop all receive/task execution on pause

```
if (event.isConnectEvent()) {   var commandToSend = ogscript.getObject('sendCommand');   ogscript.putObject('sendCommand', null);   if (c...
```

Figure 4 - Listener Settings



```
abs [contexttype=opengear]
  meta
    params
    lookup [id=hosts]
      entry [key=BlackmagicDeck1.host]
      entry [key=BlackmagicDeck1.port]
    style [id=ToggleButtons, name=ToggleButtons]
    style [id=Transport, name=Transport]
  tab
    abs [name=Playback]
      button [name=Refresh Clip List]
        task
```

Name:

ID: hosts

☐ Support multiline values

☐ ogScript Code

Lookup Entries:

Key	Value
BlackmagicDeck1.host	1.2.3.4
BlackmagicDeck1.port	9993

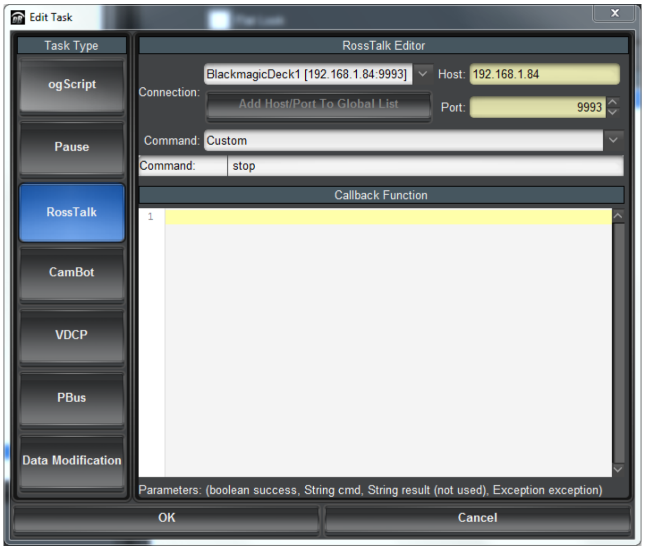
Figure 5 - Lookup: hosts

Tip

The Listener uses a "custom" setting for its connection in order to make it easy for the user to set up. The IP address is stored in a string parameter (with OID 0x9) that is accessed by this XML substitution to set the Listener's `connecthost` *attribute:* `connecthost=%value[0x9][0]%`*. When the IP address is changed, the onchange method for the OID reloads the Listener's container (the tab it's on) which, in turn, reinitializes the IP address to which it's listening. You need to access the Listener's <Source> tab and edit the XML directly to set up a Listener that behaves in this manner.*

Before looking at the Listener's code, it's instructional to describe how the connection with the deck is active.

All but one of the commands set up a TCP connection to the deck, issue a command string, and then tear the connection down again. DashBoard provides a convenient way of doing this using a feature called "RossTalk" which is fully supported in the GUI as shown in Figure 6.



Connection: BlackmagicDeck1 [192.168.1.84:9993]
Host: 192.168.1.84
Port: 9993
Command: Custom
Command: stop
Parameters: (boolean success, String cmd, String result (not used), Exception exception)

Figure 6 - RossTalk Editor

2.4.1 For Simple, One way commands, use RossTalk

To add a RossTalk command to a button, simply add a task, and then select RossTalk from the options on the left (shown highlighted blue). It's then a simple matter of selecting the recipient host from the drop down, and setting the command type to "custom". The command itself is then entered on the command line. In the example above, the command is simply "stop".

It is possible to define a callback function to handle errors and exceptions, which hasn't been done in this example.

It's also possible to use RossTalk programmatically from a script. The following example is similar to that generated by the UI in Figure 6. I've just replaced function calls that recover the IP address and port number with literal values for clarity.

```
rosstalk.sendMessage(‘192.168.1.84’, 9993, ' stop');
```

2.4.2 Use ogScript in the Listener for Transactions with the deck

The one application level transaction in this panel is for the retrieval of the clip list. The code that initiates the request is shown in Box 1, below, with the callback function in Box 2.

Recall that the Listener's ID is "commandSender" as shown in Figure 4. It is accessed programmatically by the ogScript command `getListenerById` from where we can call its start and stop methods.

Box 1 - Initializing the "Get Clips" Transaction

```
//Put the connection into a known state
ogscript.getListenerById('commandSender').stop();

//Mark ourselves as 'busy' so nothing else gets sent
ogscript.putObject('busy', true);

//Store this so the listener knows what to send
ogscript.putObject('sendCommand', 'clips get');

//Add a callback for the listener to call when we are done
ogscript.putObject('callback', callbackFunction);

//start the listener so it can send the command and read the result
ogscript.getListenerById('commandSender').start();
```

The three calls to `putObject` show how global Javascript variables can be stored and accessed within a Custom Panel. The purpose of storing the information in this manner is to allow it to be used in part of the panel that isn’t in the local scope: the Listener code.

Box 2 - The Callback Function

```
var callbackFunction = function(result) {
   //Reset our 'busy' state
   ogscript.putObject('busy', false);

   //Stop the command sender
   ogscript.getListenerById('commandSender').stop();

   //If we didn't get a result, we can't do much more
   if (result == null || result == undefined) {
      ogscript.debug('no result');
      return;
   }
   //If we did, let's update our clip list
   params.replaceConstraint(0x2, params.createIntChoiceConstraint(result));
};
```

The callback function will be invoked when the Listener has obtained a well formed response from the deck. You can see that its main purpose is to do some housekeeping, stop the listener, and update the choice constraint for the clip selector so that the user can select from an up-do-date list of clips.

2.4.3 The Listener Code

For a complete listing of the Listener code, please refer to the accompanying Custom Panel. In simple terms, there are three types of event that the Listener can act upon: connection, message and disconnection.

The code uses the `isConnectEvent`, `isMessageEvent`, and `isDisconnectEvent` methods to determine which has occurred so that it can take appropriate action.

2.4.4 `isConnectEvent`

This event occurs at the start of a TCP connection. We started one of these when the Listener's `start` method was invoked in Box 1, above.

```
var commandToSend = ogscript.getObject('sendCommand');
ogscript.putObject('sendCommand', null);
this.writeString(commandToSend + '\n', false);
```

Box 3 - Send command on isConnectEvent

So, knowing that we'd previously stored the command to send in a global variable, the code here recovers it, and sends it to the deck. Note the use of `this` to use the TCP connection, and its `writeString` method. The code shown above has been simplified for clarity.

2.4.5 `isMessageEvent`

The key Listener method used here is `event.getBytesAsString().trim()` which reads the message from the deck into a string which the rest of the code is involved in parsing. The `trim` method just removes leading and trailing whitespace from the message received.

A detailed account of the message structure and how to parse it is outside the scope of this tech note. Please look at the accompanying source code directly to understand the detail, and refer to the resources in Further Reading for the Hyperdeck protocol.

2.4.6 `isDisconnectEvent`

We don't do much here except to clear a "busy" flag.

3 Further Reading

http://www.blackmagicdesign.com/products/hyperdeckstudio

http://software.blackmagicdesign.com/HyperDeck/docs/HyperDeck_2014-06.pdf

4 Acknowledgements

Thanks to Ross Video's own Chris Kelly for putting together the initial panel, and to Matt Jefferson, Black Magic Design's Worldwide Director of Developer & Partnerships for the loan of a Hyperdeck.

5 Notices

'Blackmagic Design' is a trademark registered in the USA and other countries.

'DashBoard' and 'Panel Builder' are trademarks owned by Ross Video Ltd.