

DashBoard Beta

Version 8.3 Build: May 20, 2017

Release Notes

Copyright Notice

© 2017 Ross Video Limited. Ross® and any related marks are trademarks or registered trademarks of Ross Video Limited. All other trademarks are the property of their respective companies. PATENTS ISSUED and PENDING. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, mechanical, photocopying, recording, or otherwise, without the prior written permission of Ross Video. While every precaution has been taken in the preparation of this document, Ross Video assumes no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained herein.

Table of Contents

Version 8.3 Beta Feature Enhancements.....	3
NDI.....	3
USB Joystick Support.....	4
ogScript Additions.....	5
Version 8.2 Feature Enhancements	6
Visual Logic.....	6
Sony Camera	8
Time Picker Added	10
Date Picker Added.....	10
New Features.....	10
ogScript Features.....	10
Style Options.....	11
Bug Fixes.....	11
Version 8.1.1 Feature Enhancements	12
Ross ACID Camera	12
Panasonic Camera	12
DashBoard Revisions	12
Version 8.1 Feature Enhancements	13
Ross ACID Camera	13
Panasonic Camera	14
DashBoard Proxy Server	15
Version 8 Feature Enhancements.....	18
Visual Logic.....	18
Visual Logic Workspace.....	19
Control and APIs Panel	19
Logic Blocks	21

Version 8.3 Beta Feature Enhancements

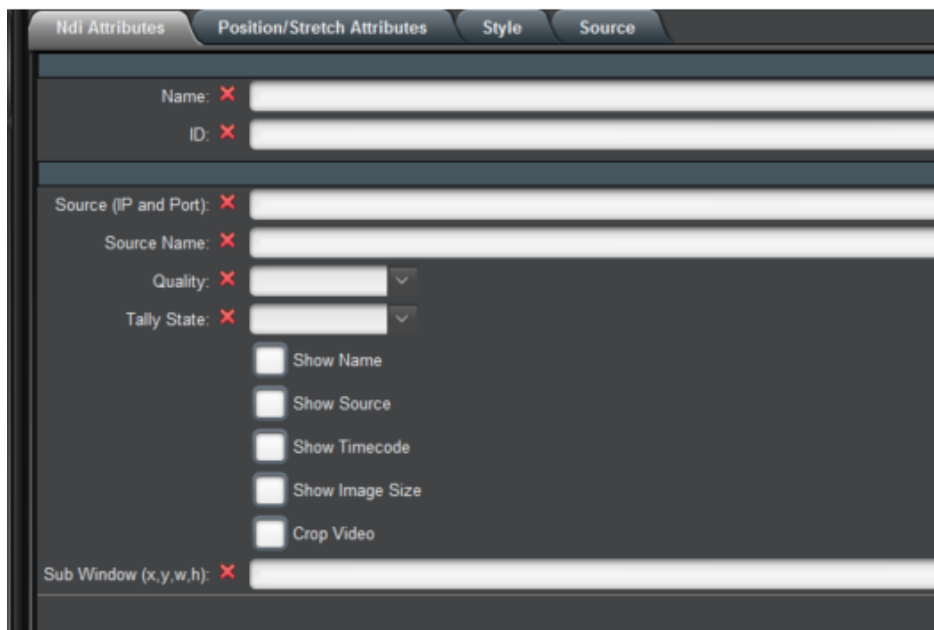
Release Date: February 22, 2016. Build May 20, 2017

This section describes new and improved functionality included in this release.

NDI

- Beta Support for NDI streams has been added. **This feature is not yet complete!**
- No editor exists yet to create this block. To add a block manually add a 'Basic Canvas Block' then manually change the tag name from 'ABS' to 'NDI'. An example block in the text editor looks like:

```
<ndi height="480" left="556" top="370" width="888"/>
```
- Double clicking on this block does now give an editor for the NDI stream, an example looks like:



The screenshot shows a configuration window for an NDI stream. It has four tabs at the top: 'Ndi Attributes', 'Position/Stretch Attributes', 'Style', and 'Source'. The 'Ndi Attributes' tab is currently selected. Below the tabs, there are several input fields, each with a red 'X' icon to its left, indicating they are required or have an error. The fields are: 'Name', 'ID', 'Source (IP and Port)', 'Source Name', 'Quality' (with a dropdown arrow), 'Tally State' (with a dropdown arrow), and 'Sub Window (x,y,w,h)'. Below the 'Tally State' field, there are five checkboxes: 'Show Name', 'Show Source', 'Show Timecode', 'Show Image Size', and 'Crop Video'.

*NDI is a trademark of NewTek corporation. More information is available at <http://NDI.newTek.com/>

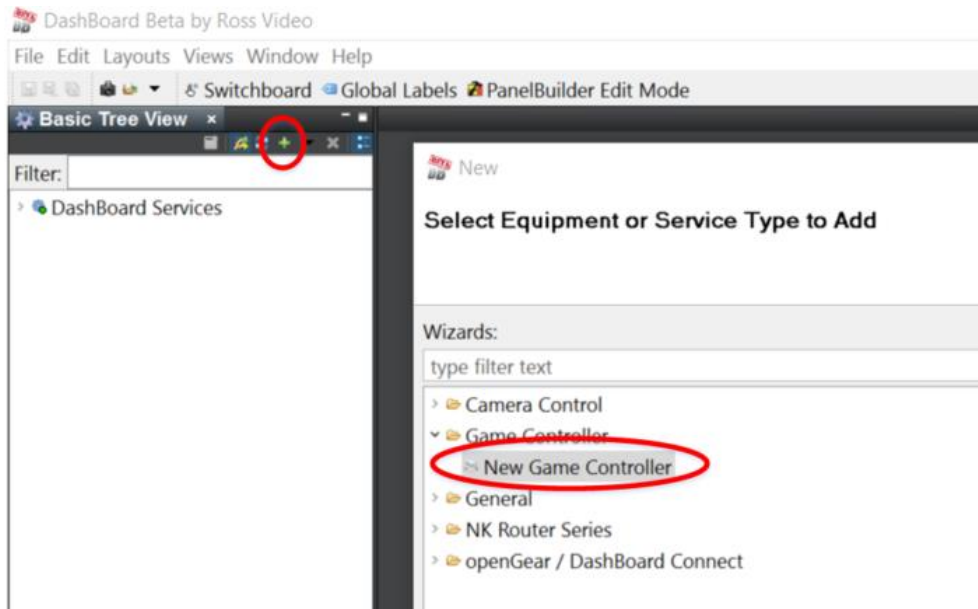
USB Joystick Support

- Support for USB joysticks has been added, in particular for the Ross Pivot Cameras.

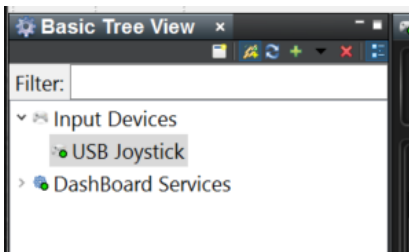
To Add a Joystick to DashBoard:

Step 1: Add a Game controller.

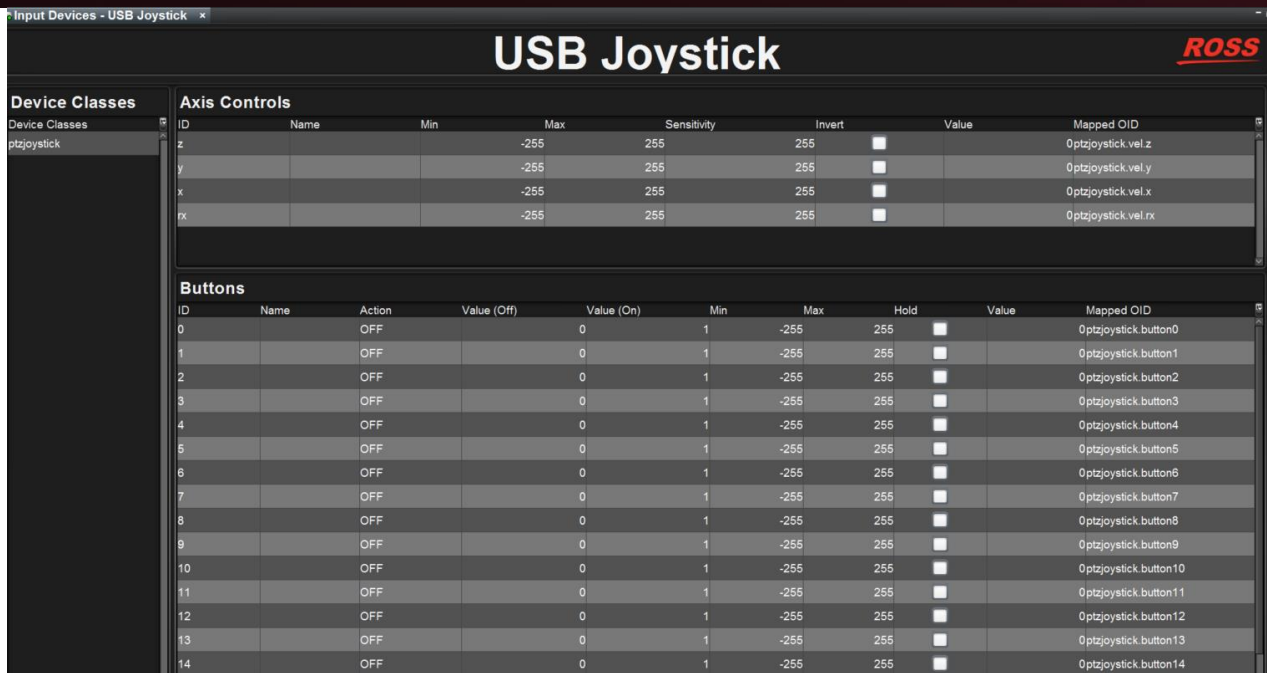
Select the '+' button in the basic tree view then select 'New Game Controller'



You can now look at the mapping USB joystick values, use the following image to see the Mappings to use by selecting the joystick in the basic tree view:



You will see something like:



The joystick is active in this mode and you can move axis and press buttons to see which item is which.

ogScript Additions

- ogscript.asyncFTPListFiles(host, port, username, password, path, callback)
 - Callback is passed success, list of files, and exception
 - file.getName()
 - file.getTimestamp() (is a java.util.Calendar object)
 - file.getSize()
 - file.isFile()
 - file.isDirectory()
- Added ogscript.asyncHTTP(URL, METHOD, CONTENT_TYPE, DATA, CALLBACK)
- Added ogscript.asyncHTTP(URL, METHOD, CONTENT_TYPE, DATA, CALLBACK, INCLUDE_RESPONSE_CODE)

DashBoard Executor Thread:

```

var asyncThread = ogscript.createAsyncExec(ID)
asyncThread.asyncExec(function)
asyncThread.asyncExec(function, delay)
asyncThread.close()
asyncThread.isClosed()
asyncThread.putWork(OBJECT) //Called from any thread - adds the object to the work queue and notifies the async thread if it
is waiting for work
asyncThread.getWork() //Called only within the asyncThread - gets work from the queue... blocks and waits if there is no work
yet
asyncThread.getWork(timeout) //Same as above except it will only block for "timeout" milliseconds
asyncThread.getWorkSize()
asyncThread.isInAsyncExec() //Returns true if the function is being executed in the asyncThread
asyncThread.takeLock() //Called from within the asyncThread
asyncThread.takeLock(Object) //Called from within the asyncThread - Take a lock and wait for a lock release with the given
object - handy for waiting for a specific response
asyncThread.releaseLock() //Called from any thread
asyncThread.releaseLock(Object) //Called from any thread - Release the lock waiting for the given object - handy for waiting

```

for a specific response

`asyncThread.waitOnLock()` //Called from the `asyncThread` to start the blocking wait

`asyncThread.waitOnLock(timeout)`

`asyncThread.waitForRelease()` //Wait for someone to call "release"

`asyncThread.waitForRelease(timeout)`

`asyncThread.release()` //Release `asyncThread` if it has called "waitForRelease"

`asyncThread.getObject(string)` //Just like `ogscript.getObject` but private to the `asyncThread`

`asyncThread.putObject(string, Object)` //Just like `ogscript.putObject` but private to the `asyncThread`

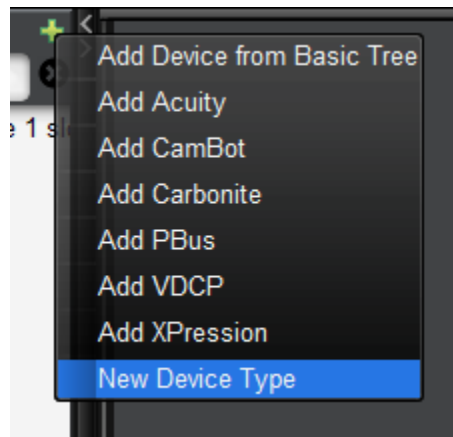
Version 8.2 Feature Enhancements

Release Date: March, 2017


This section describes new and improved functionality included in this release.

Visual Logic

- Users can create their own device APIs and use the blocks in Visual Logic. The API files can be shared with other users
- Click the Green + button (Add New Equipment) beside Devices & Parameters in the top left



- This opens a pop up window where you enter the Device Type Name and the test device name along with the IP Address and Port.

 Add new device type ✕

Device Type Name:

When you create this device type, a test device of that type will be added to this panel for you automatically. If you do not want that, leave the following items blank.

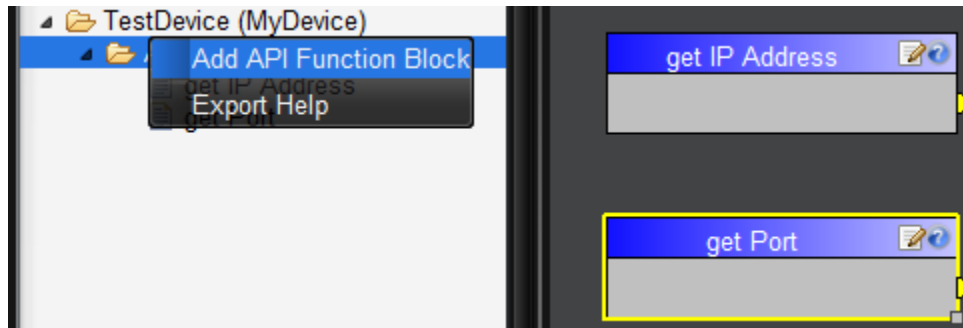
Test Device Name:

Test Device IP:

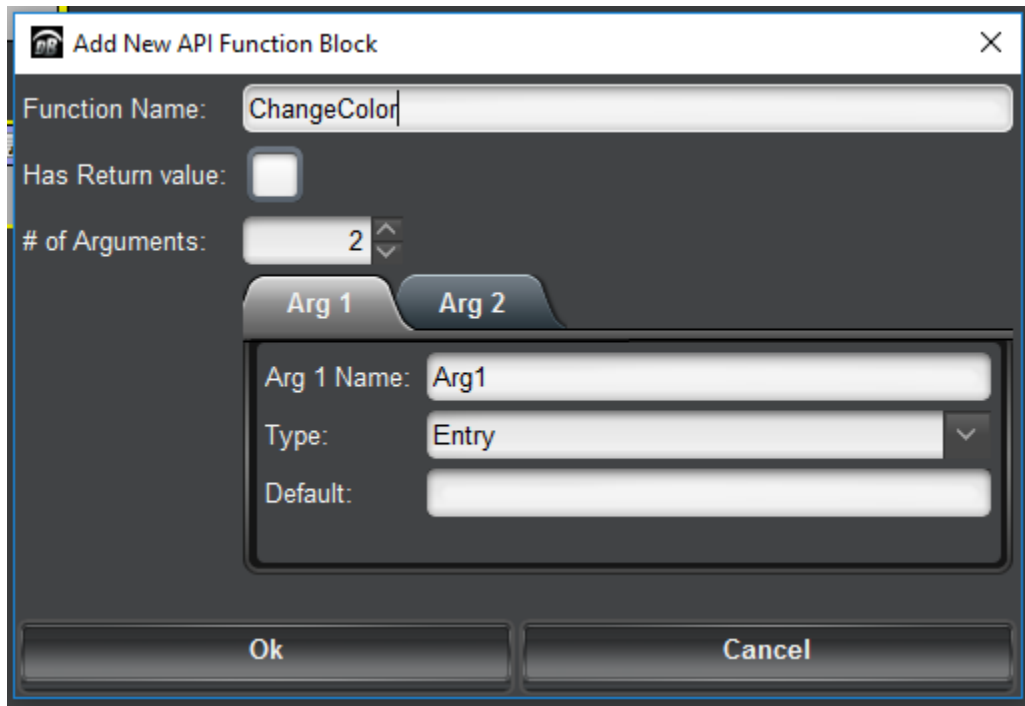
Test Device Port:

Please enter a name for your device type

- This adds a new device to the Devices & Parameters list on the left side of the interface. By default the new device has two functions: Get IP Address and Get Port. To add new API commands to the device right click on the API folder and choose "Add API Function Block".



- The Add API Function Block opens a pop up where the new function is given a name, whether it has a return value, and how many arguments are used for the command. Each argument can be named and there are four styles of argument type (Entry, Spinner, CheckBox and Dropdown). A default value can be provided.



- Once the arguments are added you are then into a Visual Logic view where you can define what the API specifically does. The new API is available in the accordion box for the device in the left hand menu.
- The API file is stored (by default) in the DashBoard directory under VisualLanguage/blocks/Devices so that file can be shared with others as desired

Sony Camera

DashBoard offers a simplified control interface for Sony cameras that are compatible with the Sony CNA-1 Camera Network Adapter.

Sony Camera Control includes support for select parameters:

- Auto White Balance and Auto Black Balance

- Master Pedestal, Red Pedestal, Green Pedestal, Blue Pedestal, Red Gain, Green Gain, Blue Gain. These are scaled to a display range of [-99, +99].
 - o The value displayed in the camera menu for these attributes may be off by one on the Sony Camera panel. This is because the CNA1 translates the received value from DashBoard's range of [-99, +99] to its range of [-32768, +32767], and then back to the camera range of [-99, +99]. The full range of values is still available; operation is not affected.
- Iris and Auto Iris
- Shutter Speed
- Camera Gain
- Loading and saving Scene Files

It should be noted that the CNA-1 has a 1:1 relationship with the camera and, as such, only one instance of DashBoard can hold a connection to the camera at a time. However, this limitation can be overcome by using a master/slave topology where one DashBoard instance will be designated as the master, or "primary camera controller" (the primary); all other DashBoard instances are then "secondary camera controllers" (the secondaries) that connect through the primary camera controller.

Once DashBoard v8.2 is installed, press the green plus button in the Basic Tree view and select "Other" to launch the "New Connection" wizard. Select Camera Control > New Sony Camera. You enter in the IP Address for the camera and choose a name for it. Press Finish and it will add the camera to the "Sony Cameras" node in the Basic Tree View.

To set up a secondary DashBoard controller to interface with the Sony camera, follow the steps below under DashBoard Proxy Server to share the Sony camera with other DashBoard instances.

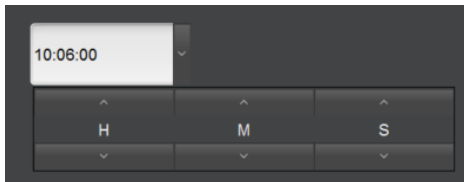
Please note: Sony camera paint control requires a license to be purchased from Ross Video.



Image 1 – Sony Camera Interface Through DashBoard

Time Picker Added

- A String can now have a Time Picker Widget created



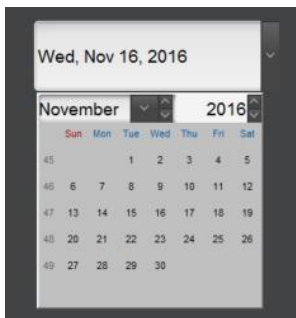
An optional formatting string can be applied using standard Java notation. (This needs to be added to the CustomPanel file by hand).

Example in the source file to show a parameter with a formatting string:

```
<param expand="true" height="73" left="1393" oid="Time.Value" showlabel="false" top="570" width="191">
  <config key="w.format">HH:mm</config>
</param>
```

Date Picker Added

- A String can now have a Time Picker Widget created ??? FINISH THIS, BUG???? ???



An optional formatting string can be applied using standard Java notation. (This needs to be added to the CustomPanel file by hand).

Example in the source file to show a parameter with a formatting string:

```
<param expand="true" height="77" left="221" oid="Date.Now" showlabel="false" top="211" width="271">
  <config key="w.format">E, MMM dd, yyyy</config>
</param>
```

Currently in Version 8.3 Beta a Data parameter needs a default value and a format string for it to appear and work correctly. The string Shown above "Wed, Nov 16, 2016" works as a default value as an example.

New Features

- Support to auto-wire Custom Panel files to USB-based devices such as paint boxes, joysticks, and selector boxes
- Support for -fullscreen command line/shortcut argument to always launch DashBoard in full screen mode.

ogScript Features

- New ogScript function to work with RESTFUL APIs
ogscript.asyncHTTP(URL, METHOD, CONTENT_TYPE, DATA, CALLBACK)

- Ability to enable/disable buttons and other controls with ogScript.setEnabled(COMPONENT_ID, TRUE/FALSE)

Style Options

- New options for round style for DashBoard custom panels
- Support for the “nudge” function for both pressed and unpressed states of buttons (previously only supported through by toggle buttons)
- Localization of DashBoard custom panels is now possible with src attribute lookup (lookup to load a Java XML or standard properties file)
- Inset cell padding in a table parameter (previously done cell-by-cell in table but this is more efficient)
- lockchildaspects attribute inside of <simplegrid/> which prevents distorting internal components by stretching them to different aspect ratios

Bug Fixes

- Improvements to the way scrolling works with mouse wheel
 - Made autowire more robust for custom panels when DashBoard is launched
 - Bug fix that made it difficult to remove the final style element from on/off style tags
 - Fix bug where JSON updates to read-only state could remove style information from spinners
 - Fix bug where stack overflow errors thrown in ogScript could disable running additional tasks until panel is reloaded
 - Improvements to performance when repainting the DashBoard UI to allow timers and fade effects to run smoothly
- Removed Undo/Redo function from Visual Logic
- [Fix bug in table parameters that can cause certain tables to stop updating correctly after adding/removing numerous rows](#)

Version 8.1.1 Feature Enhancements

Release Date: September 1, 2016

This section describes improved functionality included in this release.

Ross ACID Camera

- Revised ACID Camera UI's "Camera Server" node so that it cannot be removed while other cameras exist

Panasonic Camera

- Revised Scene File to boot up in the proper state
- Cameras can be reconnected from both the primary and secondary systems after being disconnected (previously the cameras could only be reconnected to the primary system)
- Malformed messages are handled more efficiently when received from the Panasonic camera
- Modifications to the menu lock functionality

DashBoard Revisions

- Change == check in onchange handler for parameters back to .equals
 - Allow .equals check/filter to be removed with alwaystrigger="true"
- Add configuration options to level meter
 - w.orientation = horizontal/vertical
 - w.reverse = true/false (change from bottom/up to top/down or left/right to right/left)
 - w.redcolor = color for red LEDs
 - w.yellowcolor = color for yellow LEDs
 - w.greencolor = color for green LEDs
 - w.redcount = number of red LEDs
 - w.yellowcount = number of yellow LEDs
 - w.greencount = number of green LEDs
- Update table UI in device mapping selector when device status changes

Version 8.1 Feature Enhancements

Release Date: July 11, 2016

This section describes new and improved features included in this release.

DashBoard Version 8.1 adds Camera Control for both Ross ACID Cameras and Panasonic cameras. This allows users to connect to the camera directly from DashBoard as a device as opposed to creating a custom panel.

Ross ACID Camera

DashBoard is the control interface for the Ross ACID Cameras. Once the camera is connected as a DashBoard device full control of the CCU is served from the camera and can be modified through the interface.

ACID Cameras have been designed for maximum HD performance in any studio production environment. They offer best-in-class resolution, sensitivity and signal to noise ratio, plus unique UltrachromeHR outputs for chroma key applications.

Once DashBoardv8.1 is installed, press the green plus button in the Basic Tree view and select “Other” to launch the “New Connection” wizard. Select Camera Control > New ACID Camera. You enter in the IP Address for the camera and choose a name for it. Press Finish and it will add the camera to the “Ross Video ACID Cameras” node in the Basic Tree View.

The camera node in the Basic Tree View offers a compact “Basic Controls” user interface or a more complete camera paint control set with the “Remote Control” user interface (shown below).



Image 2 – ACID Camera CCU Interface Through DashBoard

Panasonic Camera

DashBoard can directly control two models of serial Panasonic cameras, specifically the AK-HC1500G and AK-HC1800N. As DashBoard speaks IP, the commands need to be sent through a Control IP-to-Serial bridge device. The Control device has four ports so four cameras can be connected to each and the camera protocol needs to be set to 3.

It should be noted that, as the Panasonic cameras are serial devices, only one instance of DashBoard can hold a connection to the camera at a time. However, this limitation can be overcome by using a master/slave topology where one DashBoard instance will be designated as the master, or “primary camera controller” (the primary); all other DashBoard instances are then “secondary camera controllers” (the secondaries) that connect through the primary camera controller.

Once DashBoard v8.1 is installed, press the green plus button in the Basic Tree view and select “Other” to launch the “New Connection” wizard. Select Camera Control > New Panasonic Camera. You enter in the IP Address for the camera and choose a name for it. Press Finish and it will add the camera to the “Panasonic Cameras” node in the Basic Tree View.

To set up a secondary DashBoard controller to interface with the Panasonic camera, follow the steps below under DashBoard Proxy Server to share the Panasonic camera with other DashBoard instances.

Please note: Panasonic camera paint control requires a license to be purchased from Ross Video.

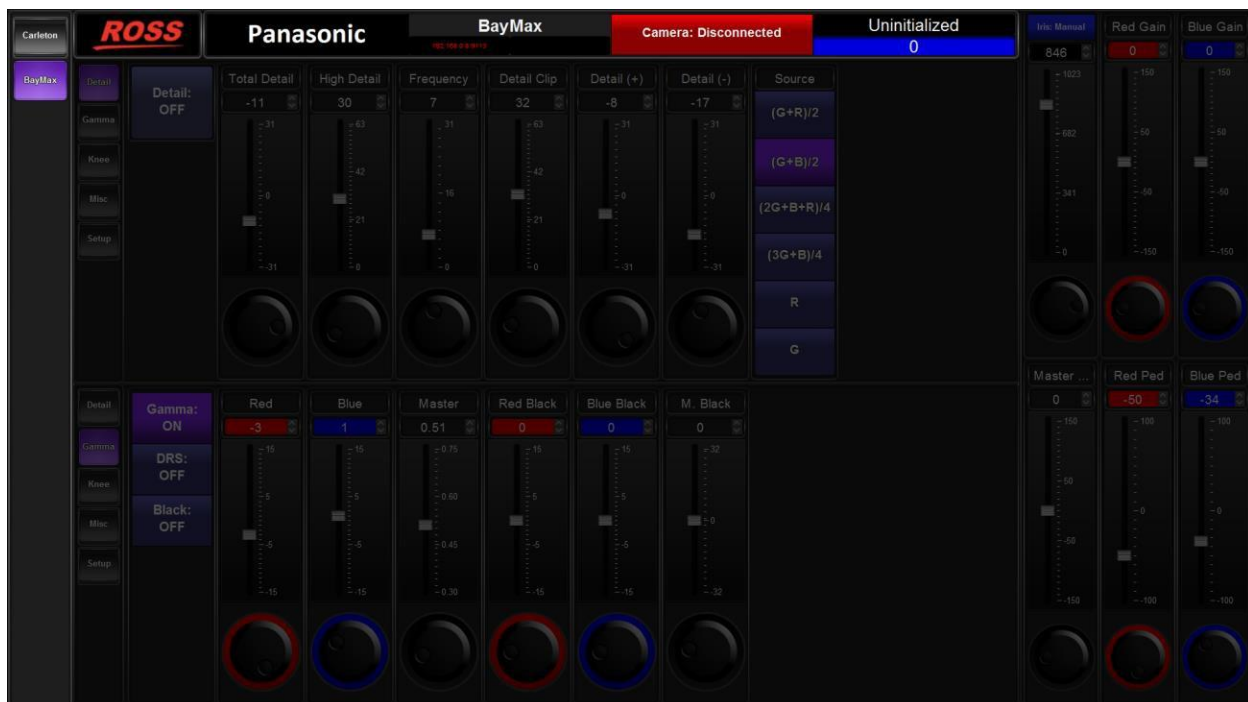


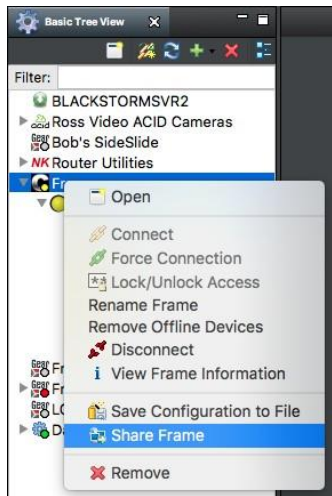
Image 3 – Panasonic Camera User Interface in DashBoard

DashBoard Proxy Server

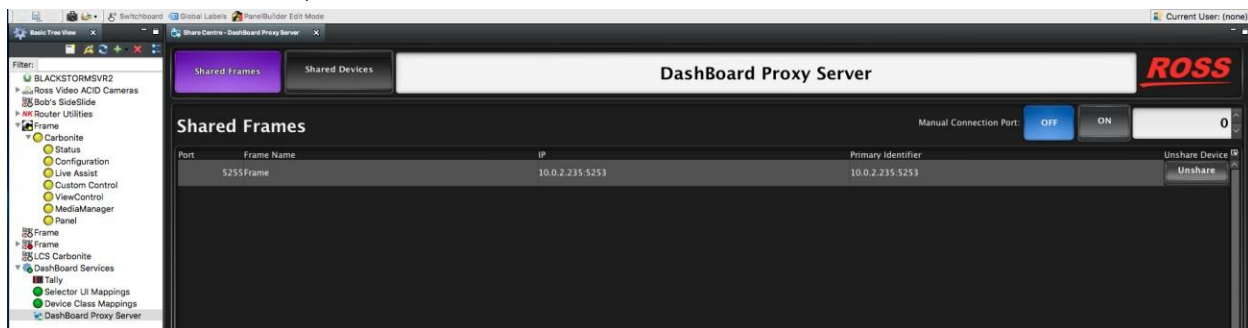
The DashBoard Proxy Server is a way to connect to devices from distant locations efficiently. Without using a DashBoard Proxy Server it can sometimes take a very long time to populate the DashBoard tree and get access to devices. Instead, the Proxy Server will operate at the remote facility and provide faster access to openGear products.

To set up a DashBoard Proxy Server follow these steps:

- Install DashBoard v8.1
- Add all of the local equipment to the DashBoard Basic Tree View as per normal (go to File -> New -> TCP/IP DashBoard Connect or openGear Device if they do not connect automatically)
- Right click on the frame you want to share and select "Share Frame"



- In the DashBoard Services tree node, open the DashBoard Proxy Server (you should see your shared frame listed)

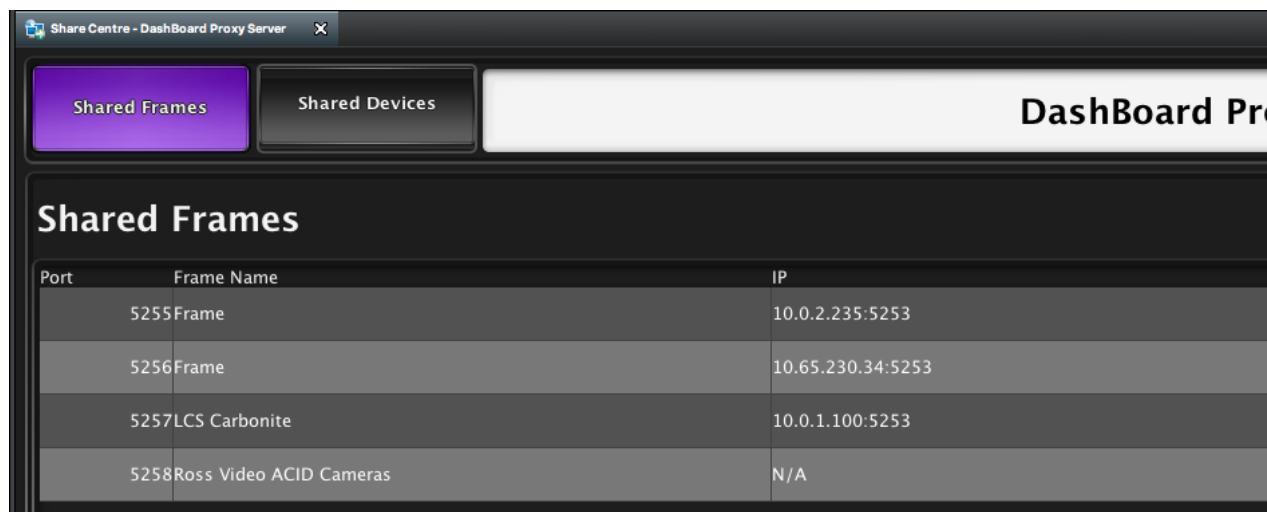


- Set the "Manual Connection Port" to an open port on your computer so DashBoard can fetch your shared frame information over HTTP (recommend port 80 or 8080)
- This will make it easy to add all of the frames from proxy server at once and under a single node in the DashBoard Basic Tree View



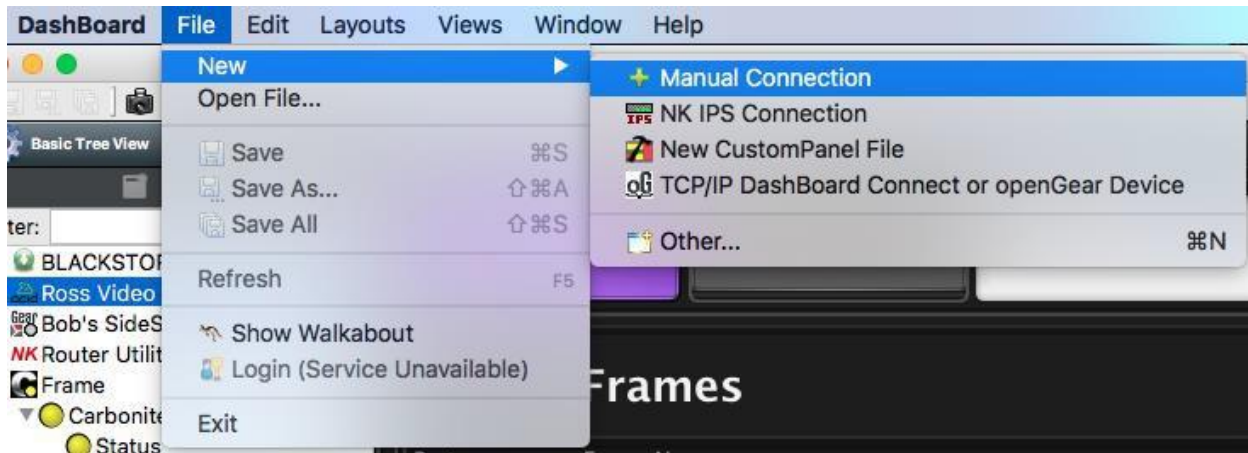
- Share any additional frames by repeating to right-click on them and selecting the “Share Frame” option.

Note: Your firewalls will need to allow access to both the “Manual Connection Port” and the “Shared Frame Ports” (the right-most column in the table – 5255, 5256, 5257, and 5258 in this example)

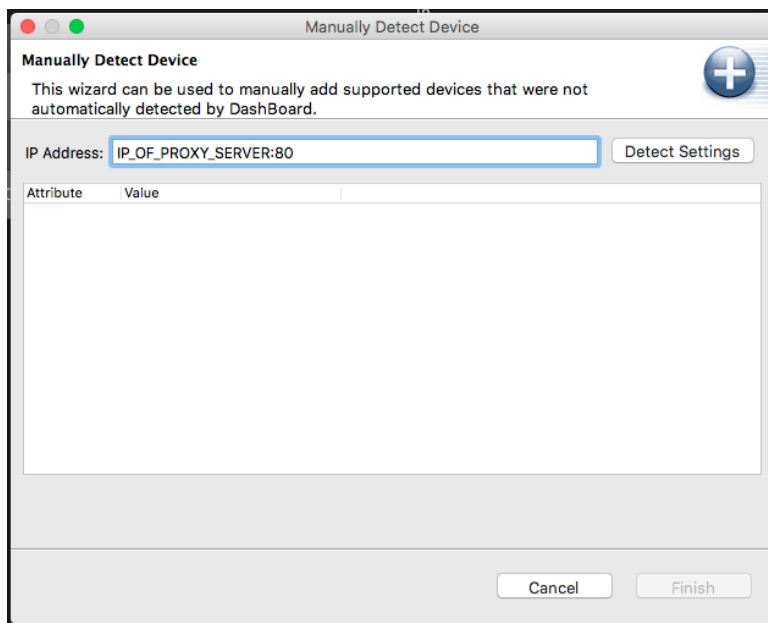


To set up a DashBoard Client PC follow these steps:

- Install DashBoard v8.1
- Got to File -> New -> Manual Connection



- Enter the IP of the DashBoard Proxy Server followed by a colon and the port you specified as the "Manual Connection Port" (80 in this example)



- Select "Detect Settings" and verify that the proxy server's information is detected before hitting "Finish"
- A new node appears in your Basic Tree View called "DashBoard Proxy Server" and contains the shared frames underneath of it.

DashBoard Proxy Server Notes:

- If you do not want to share everything, you can share a subset of devices by right clicking on the individual devices and selecting "Share Device". They will appear in the "Shared Devices" tab of the proxy server configuration screen.
- You can connect to the frames individually instead of all at once by going to File -> New -> TCP/IP DashBoard Connect or openGear Device, entering the proxy server's IP address, the port from the Shared Frames table, and selecting "JSON" as the protocol.

Version 8 Feature Enhancements

Release Date: April 13, 2016

This section describes new and improved features included in this release.

DashBoard Version 8 adds in a lot of features to make developing Custom Panels easier and providing new features to get the most out of your panels. DashBoard version 8 takes the powerful functionality of the DashBoard platform and adds the ability to create Custom Panels using Visual Logic. This allows users to create fully functional panels without ever writing a single line of JavaScript code.

Visual Logic

DashBoard Visual Logic is a visually-oriented code authoring environment that enables you to quickly create and edit segments of ogScript code for your CustomPanels. Visual Logic is part of Panel Builder.

ogScript is a JavaScript-based programming language designed to interact with DashBoard-enabled devices. In DashBoard CustomPanels, you can use ogScript to define interactions between panel objects, and to communicate with external devices. You can create and edit ogScript manually, or use Visual Logic to create and edit it visually.

Visual Logic enables CustomPanel creators who have limited JavaScript experience to more easily add ogScript functionality and logic to CustomPanels. In the Visual Logic editor, you drag pre-made logic blocks into the workspace, and then connect them to define their logical relationships. PanelBuilder creates the underlying ogScript code for you.

Tip: The DashBoard Visual Logic editor is similar to the Visual Logic editor in Ross Video XPression, so if you learn to use one, you can easily learn to use the other.

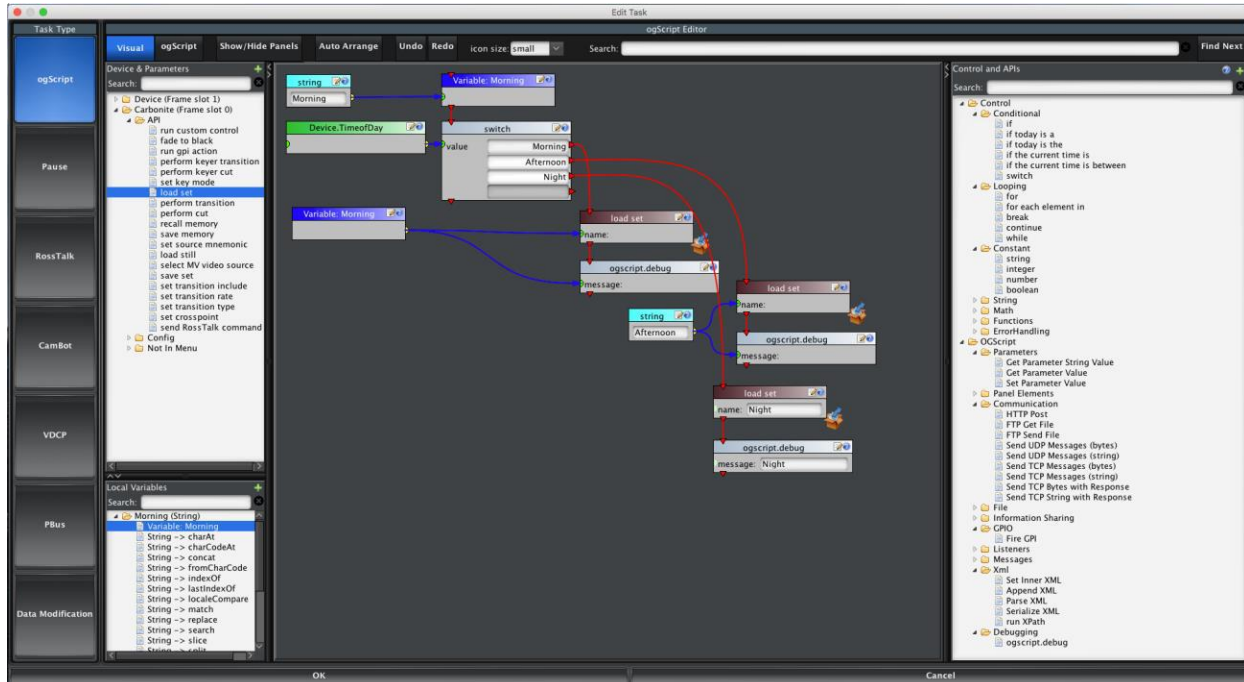


Image 1 – DashBoard v8 Visual Logic View

Visual Logic Workspace

The central area of the Visual Logic editor is the workspace. This is where you drag in objects (such as parameters, variables, and functions) to create logic blocks, and then link the blocks to establish logical connections between them.

If the segment of ogScript code you are editing has multiple functions, each function appears on a separate tab in the Visual Logic workspace.

On the left hand side are a list of the device and parameters that have been added to the DashBoard panel. New devices can be added by pressing the green plus button which allows you to enter in a name, IP Address, and assign a color to blocks associated with that device. On the lower left is a list of local variables that can be created that are accessible by that specific task.

Tip: The workspace is usually larger than the available display space. Use the scroll bars on the right and bottom of the workspace to adjust the view.

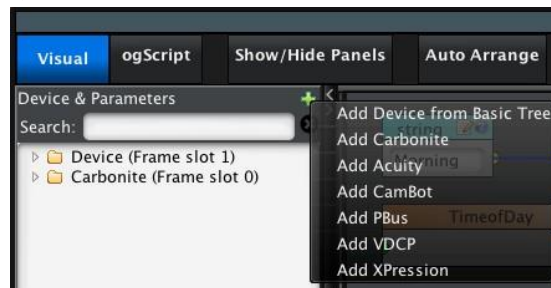


Image 2 – DashBoard v8 Adding a Device

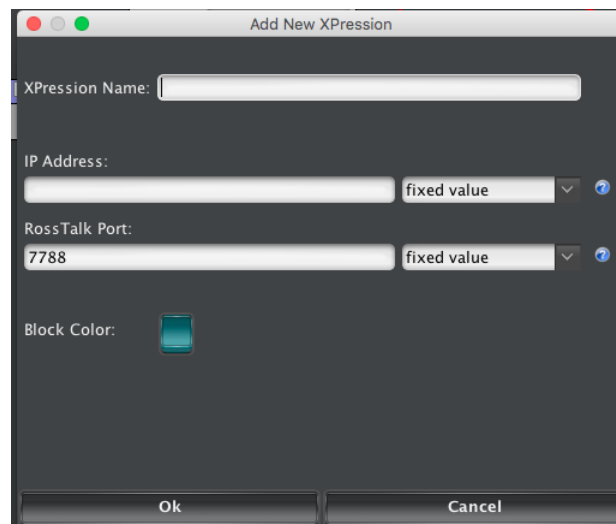


Image 3 – DashBoard v8 Defining a Device Example

Control and APIs Panel

The Control and APIs panel lists logic blocks associated with logical operations (controls) and API functions (including ogScript functions) on the right hand side of the interface. Additional APIs can be created by the user to control 3rd party IP-based devices if needed. There is a search bar in the upper right to quickly find specific commands of interest.

Control:

The Control folder contains logic blocks that perform logical and mathematical operations. You can use these logic blocks to test conditions (if, switch), set up loops (for, while), parse and manipulate string data, and perform mathematical calculations.

ogScript:

The ogScript folder contains ogScript functions that can get/set parameter values, manipulate panel elements, read and write to files, read/write/parse messages, communicate by HTTP, FTP, UDP, and TCP/IP, and more.

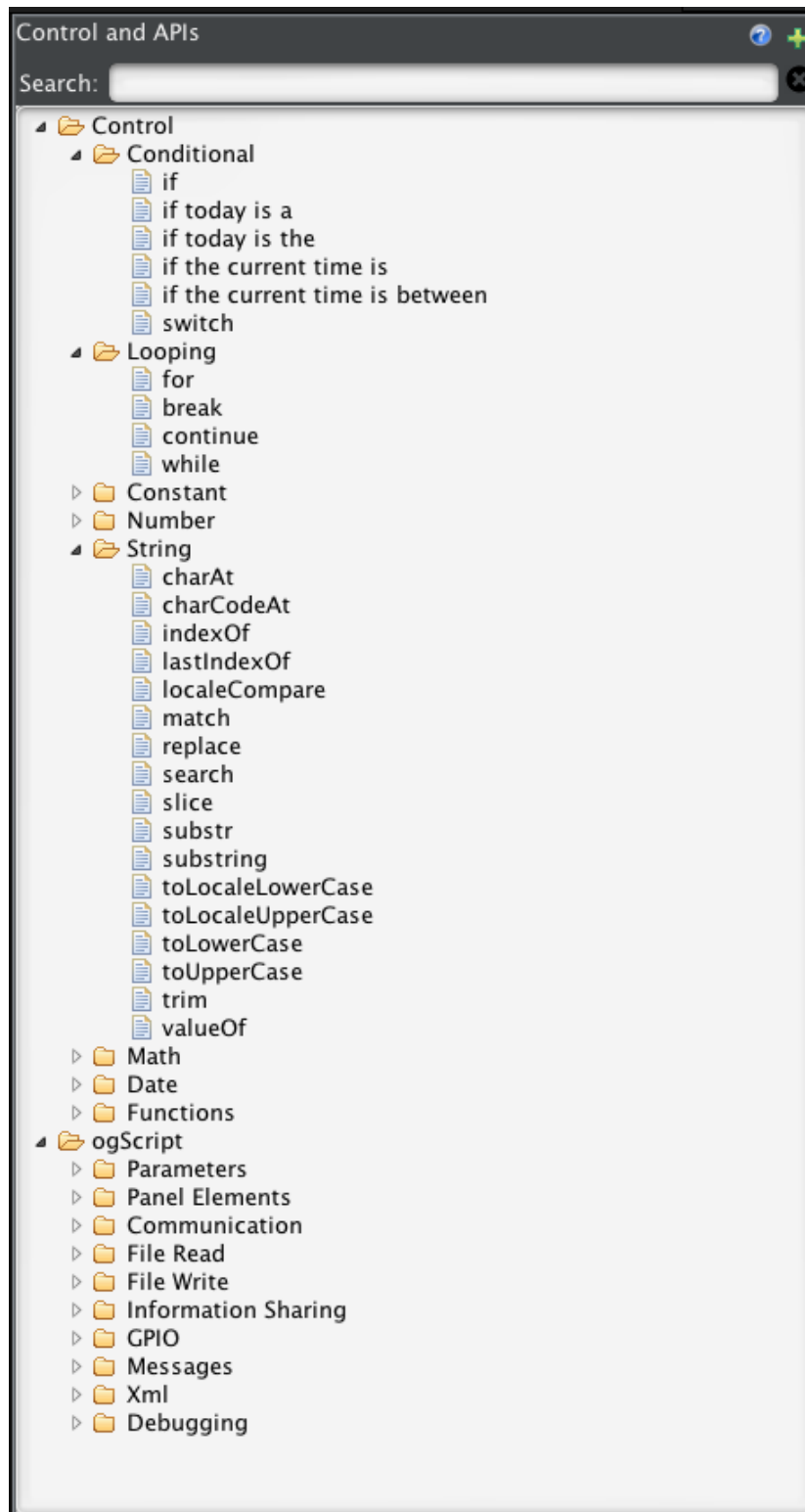
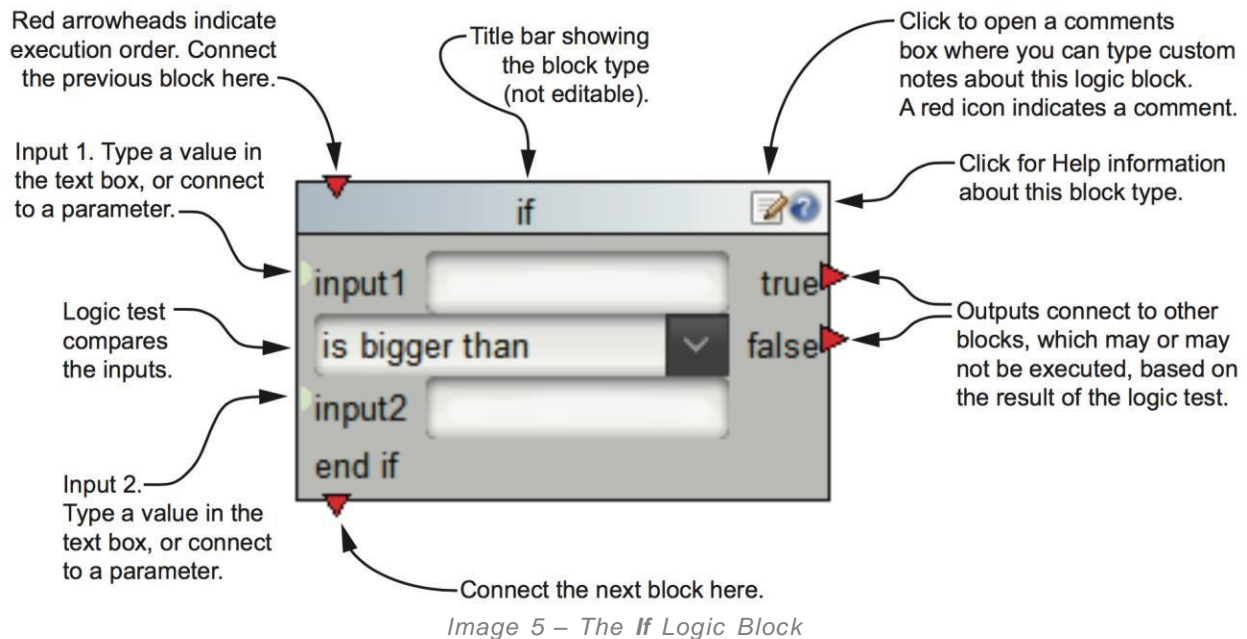


Image 4 – DashBoard v8 Control and ogScript Commands

Logic Blocks

Any of these blocks can be dropped into the Visual Logic area to provide a visual representation. . Each logic block represents a functional unit, such as a parameter, a local variable, a logical control, or an ogScript function. To create a working ogScript code segment, you drag multiple logic blocks into the workspace and then link them together to define how they interact.



To view the ogScript press the ogScript button at the top of the panel beside the Visual button. Please note that changing the ogScript directly will prevent the Visual Logic window from being available for that specific task (but other tasks will still have access to the Visual Logic view). The Visual Logic interface can be automatically arranged by pressing the Auto Arrange button at the top of the panel to clean up the view.

