DashBoard

Version 8.2 BETA

Release Notes

Table of Contents

www.rossvideo.com • Phone: +1 613-652-4886 • Technical Support: +1 613-652-4886

Version 8. 2 Features

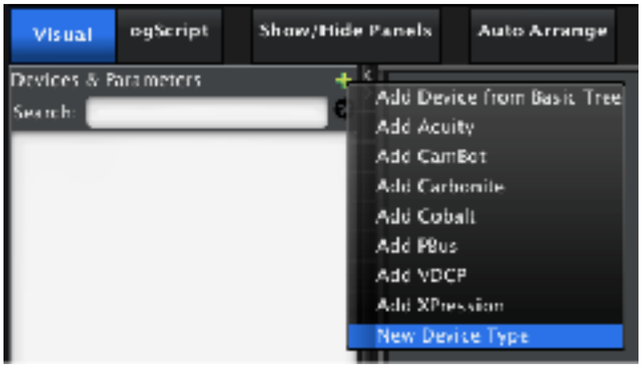
BETA Release Date: November 4, 2016

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

Adding a Custom Device API in Visual Logic

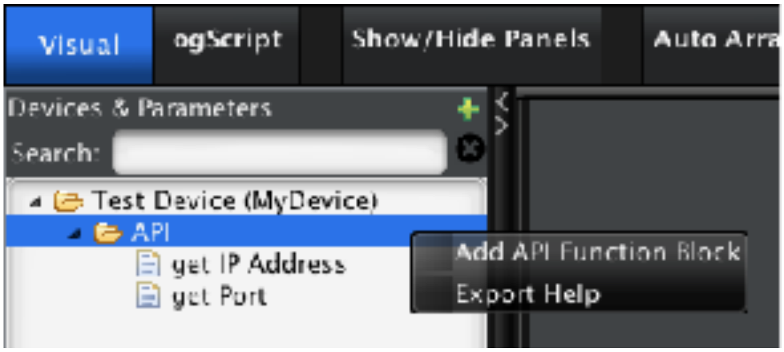
While there are a number of devices provided in the Visual Logic function of DashBoard, there may be instances where another device needs to be added. With this new feature, users can create their own API blocks for a custom device.

First, enter the Visual Logic editor in the usual manner. In the top left pane, press the green + button to get a drop down list of available devices, and choose the "New Device Type" option.

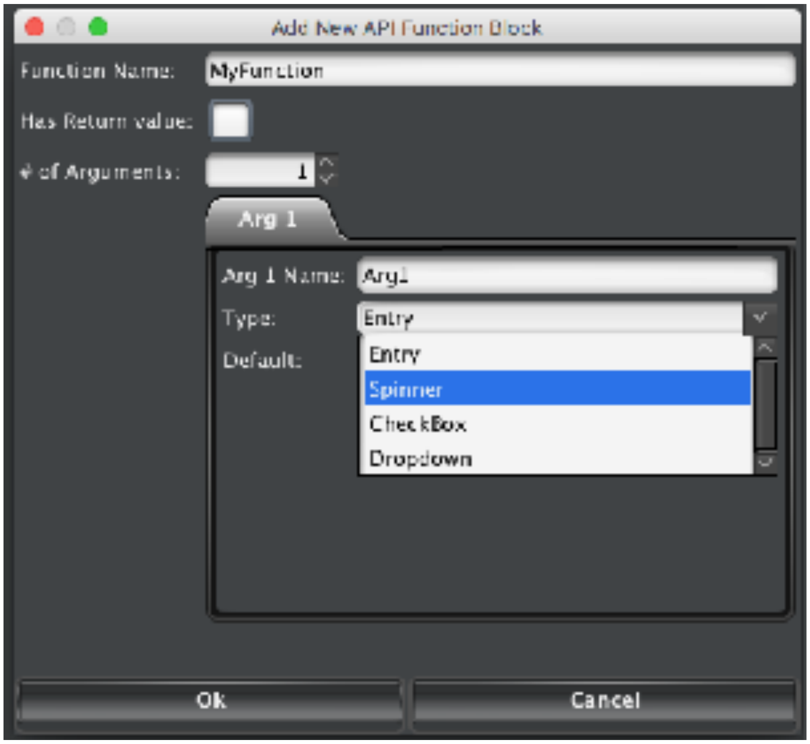


You can name the device, as well as set up a test device that you are interfacing with for ensuring the API is set up properly by putting in a test device name, IP address and port.

This creates the device in the Devices & Parameters pane. By default this will have two items in the API accordion box: Get IP Address and Get Port. To add a new API block, right click on the API folder and select "Add API Function Block"



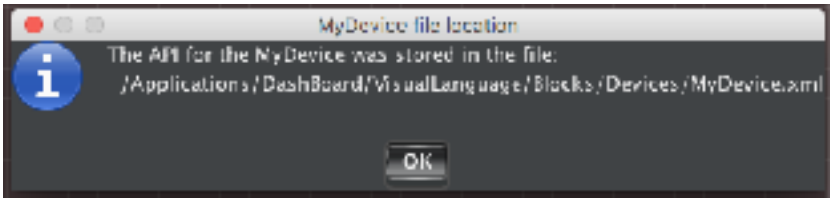
When chosen, a pop up window will appear with entries for naming the function, whether or not it has a return value, and the number of arguments (each argument creates an Arg X tab where you can name the argument, choose the type, and provide a default value).



By pressing Ok the pop up window will close and a new tab will appear in the Visual Logic editor for the function where the command can be created using Visual Logic Control and APIs from the right hand pane or typing the code directly (although that will lock out the Visual Logic aspect for inputting commands for the function).

Expanding the newly created device and API folders in the Devices & Parameters pane will reveal the new API block that was created, which can be dropped into the Visual Logic editor just like other devices offer.

Pressing Ok in the Visual Logic editor will result in a pop up window advising you as to where the API file has been stored. The file can be copied and sent to other DashBoard instances so that the API is available on that copy of DashBoard.


MyDevice file location
The API for the MyDevice was stored in the file:
/Applications/DashBoard/VisualLanguage/Blocks/Devices/MyDevice.xml
OK

Version 8.1.2 Feature Enhancements

Release Date: October 17, 2016

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

DashBoard Improvements

- Extended the timeout for fetching device OGLML documents
- Revised touch wheel UI to correctly calculate the number of revolutions to reach the maximum value
- The color picker now has a w.showstring option to disable showing the text for the string
- Numerous improvements for Structure parameters
- Added vdcp.activeClipInfo (host, port, channel, callback) which calls back

    getClipDuration
    getClipDurationAsString
    getClipID
    getClipPosition
    getClipPositionAsString

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 Improvements

- 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.grencount = 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 you will want to press the green plus button in the Basic Tree view and select Camera Control > New 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 Basic Tree view on the upper left hand side pane.

You will have a new accordion list that has items you can open (Basic Controls and Remote Control) providing the CCU.

Image 1 – ACID Camera CCU Interface Through DashBoard

Panasonic Camera

DashBoard is able to 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 Comtrol IP-to-Serial bridge device. The Comtrol 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 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 DashBoardv8.1 is installed you will want to press the green plus button in the Basic Tree view and select Other, select Panasonic Camera from the list and press Next. You enter in the IP Address and Port for the Comtrol IP to serial converter and choose a name for the Panasonic camera. Press Finish and it will add the camera to the Basic Tree view on the upper left hand side pane. This sets up the camera for the Primary Camera Controller.

To set up a secondary DashBoard controller to interface with the Panasonic camera you go to the Primary Camera Controller and find the camera in the Device Tree and right click on it. This brings up a menu where you can choose the Share Device option. The devices that are shared appear in the DashBoard Services -> DashBoard Proxy Server items in the device tree.

On the Secondary DashBoard Controller go to File -> New -> TCP/IP DashBoard Connect or openGear Device. Enter the IP Address of the Primary Camera Controller and select JSON as the protocol. You can also choose a name for the Panasonic camera for the Secondary DashBoard Controller. A new category will appear in the device tree.

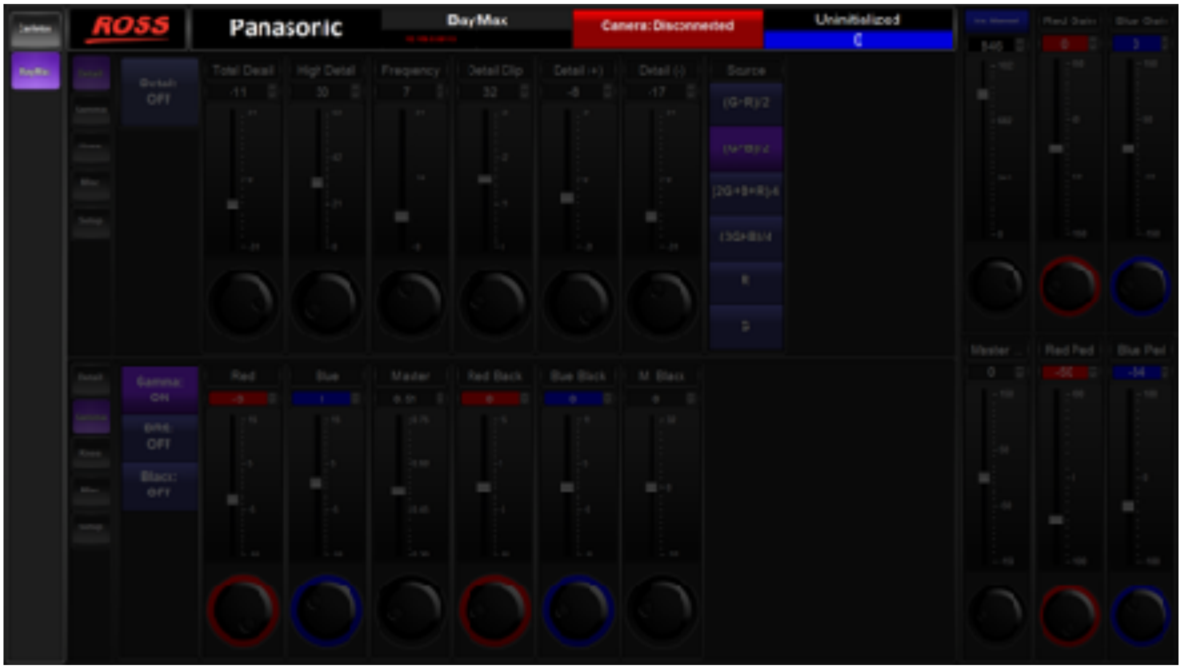


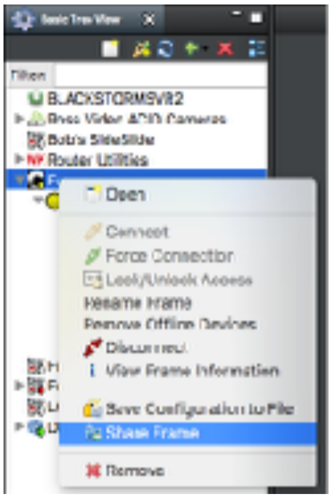
Image 2 – Panasonic Camera User Interface in DashBoard

DashBoard Proxy Server

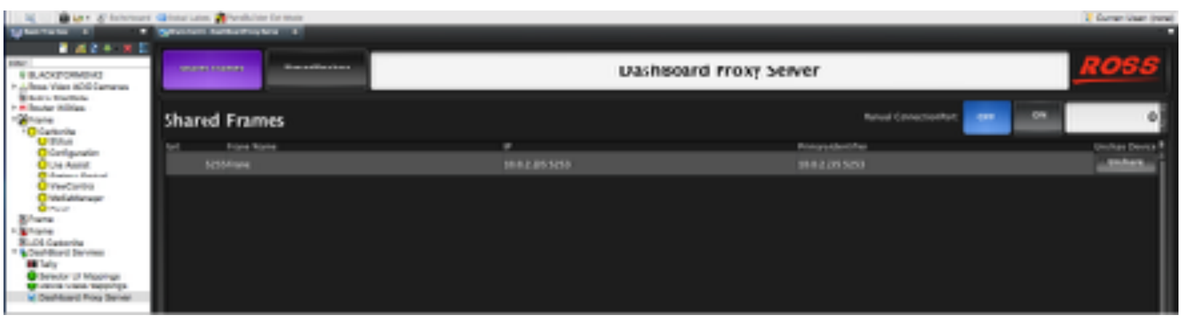
The DashBoard Proxy Server is a way to connect to devices from distant locations efficiently. Without using a DashBoard Proxy Server it can sometimes take a very long time to populate the DashBoard tree and get access to devices. Instead, the Proxy Server will operate at the remote facility and provide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

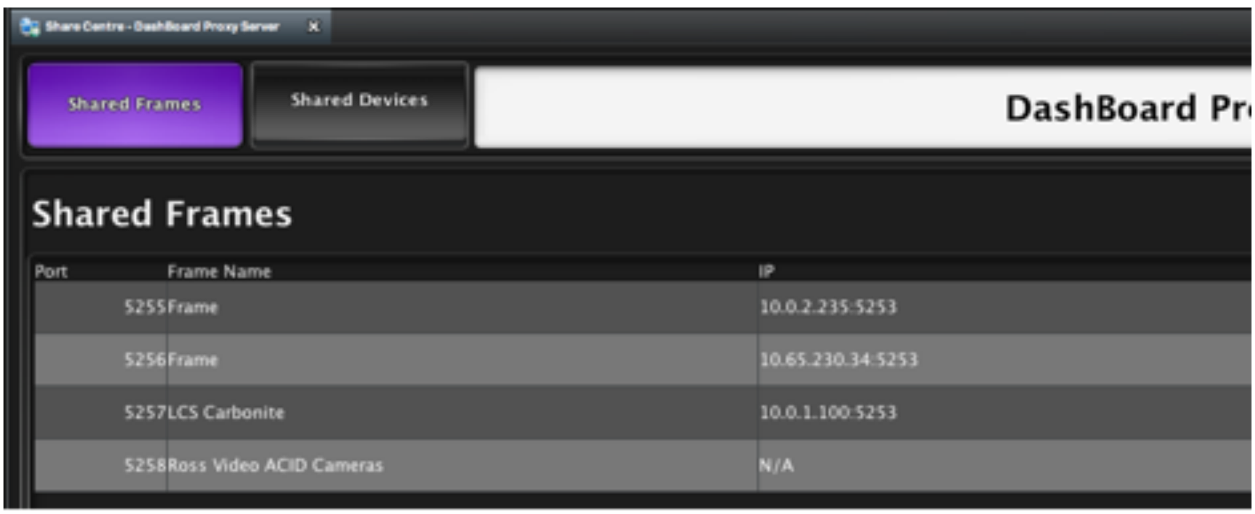


- Set the "Manual Connection Port" to an open port on your computer so DashBoard can fetch your shared frame information over HTTP (recommend port 80 or 8080)
- This will make it easy to add all of the frames from proxy server at once and under a single node in the DashBoard Basic Tree View

www.rossvideo.com • Phone: +1 613-652-4886 • Technical Support: +1 613-652-4886

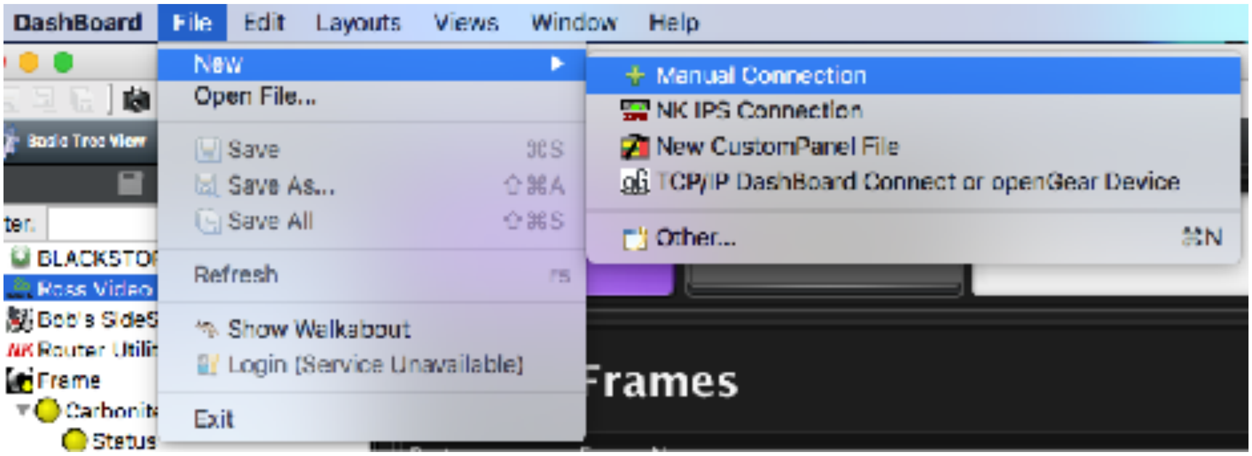
- Share any additional frames by repeating to right-click on them and selecting the “Share Frame” option.

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)

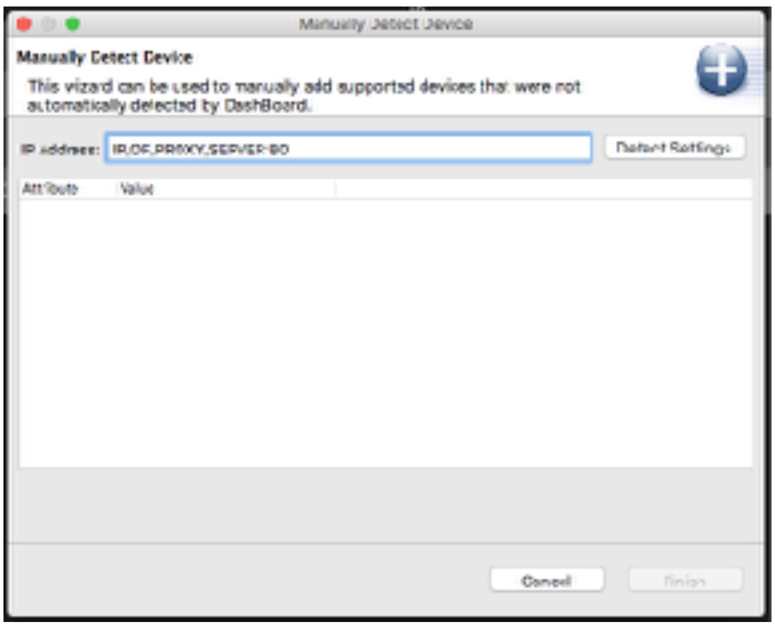


To set up a DashBoard Client PC follow these steps:

- 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.

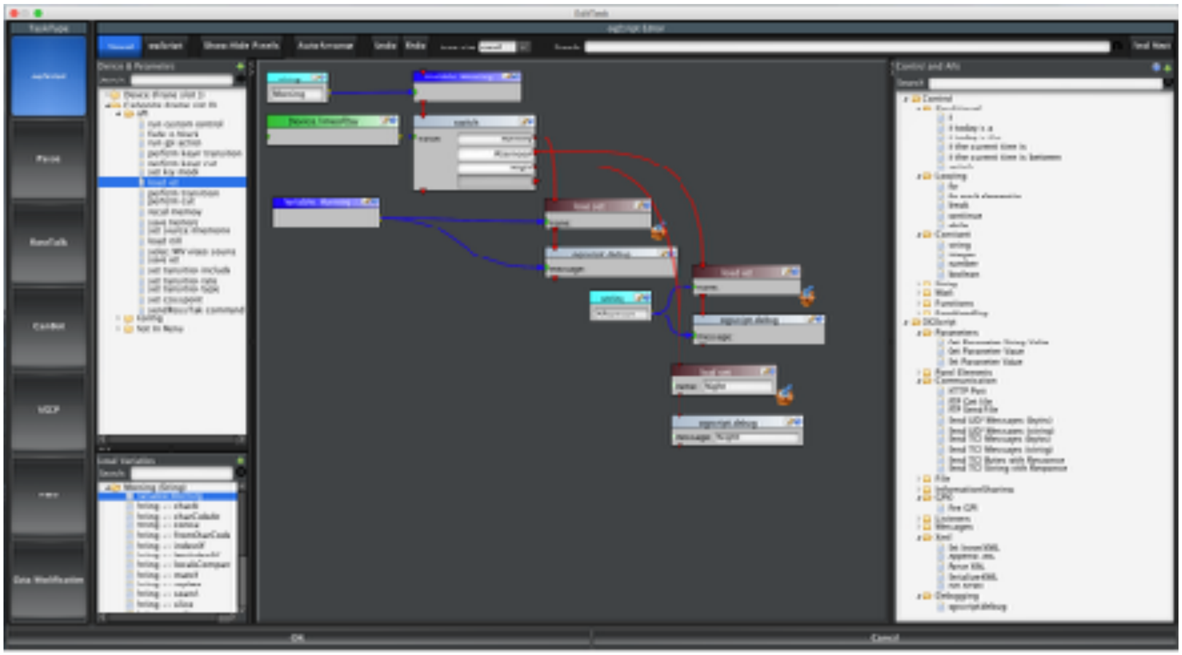


Image 1 – DashBoard v8 Visual Logic View

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.

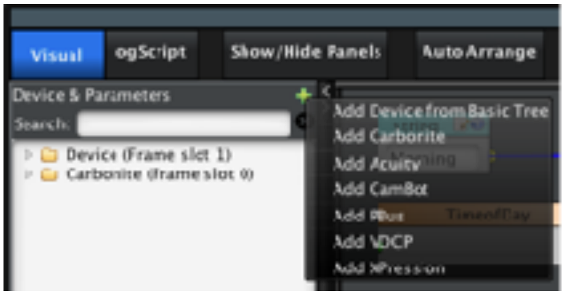


Image 2 – DashBoard v8 Adding a Device

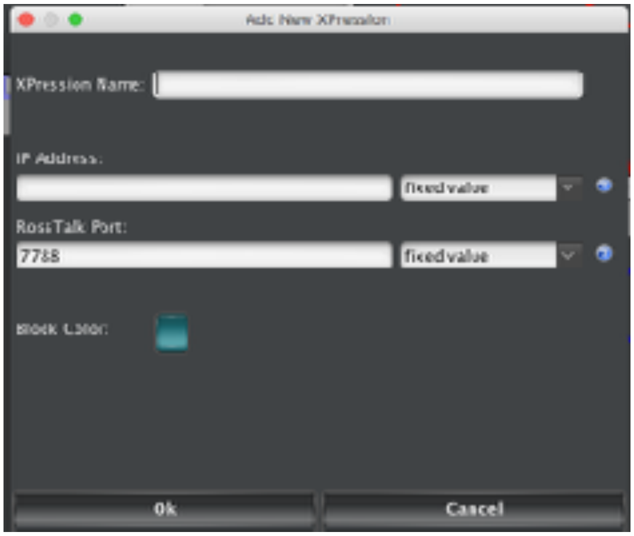


Image 3 – DashBoard v8 Defining a Device Example

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.

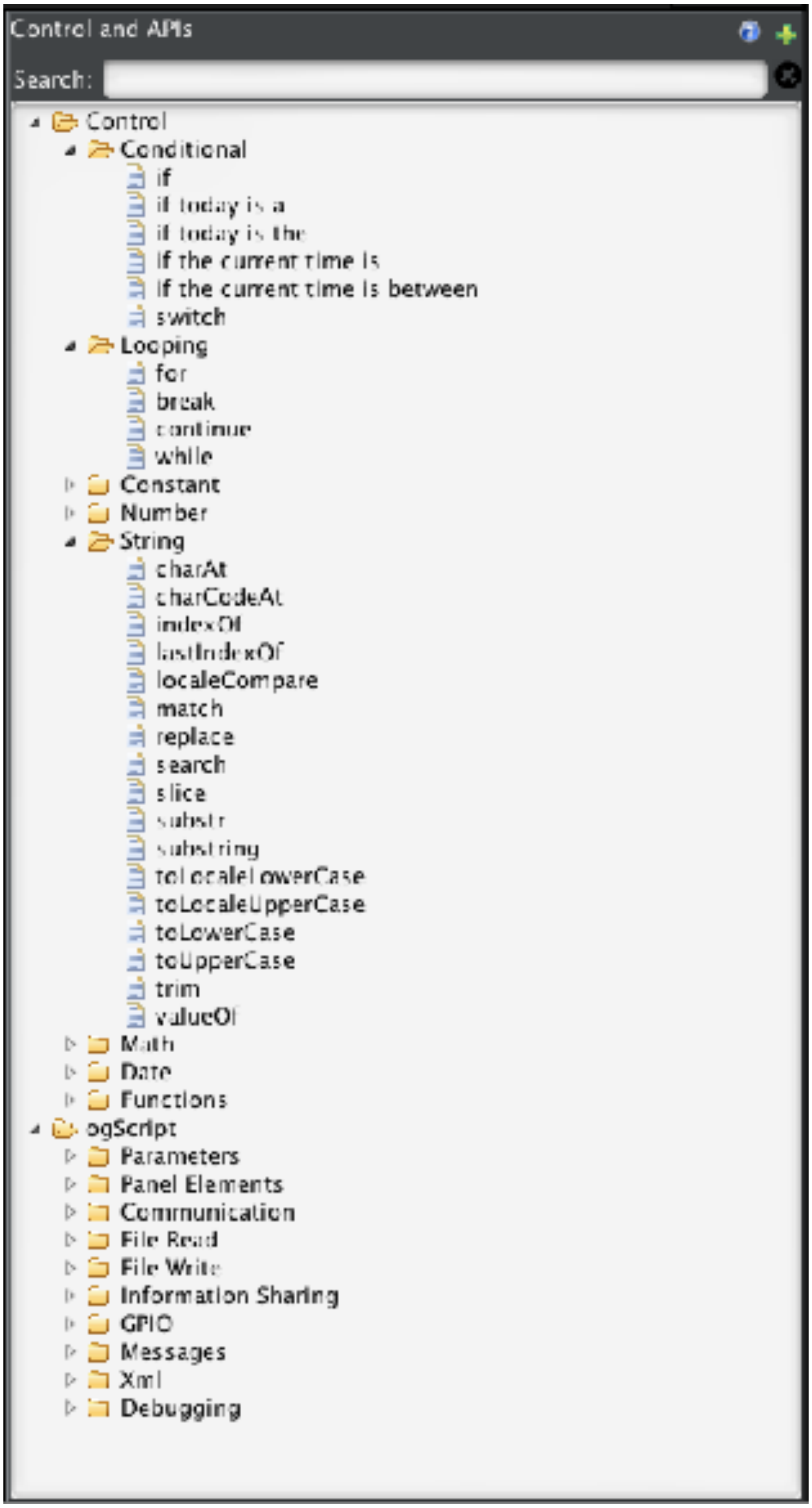


Image 4 – DashBoard v8 Control and ogScript Commands

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.

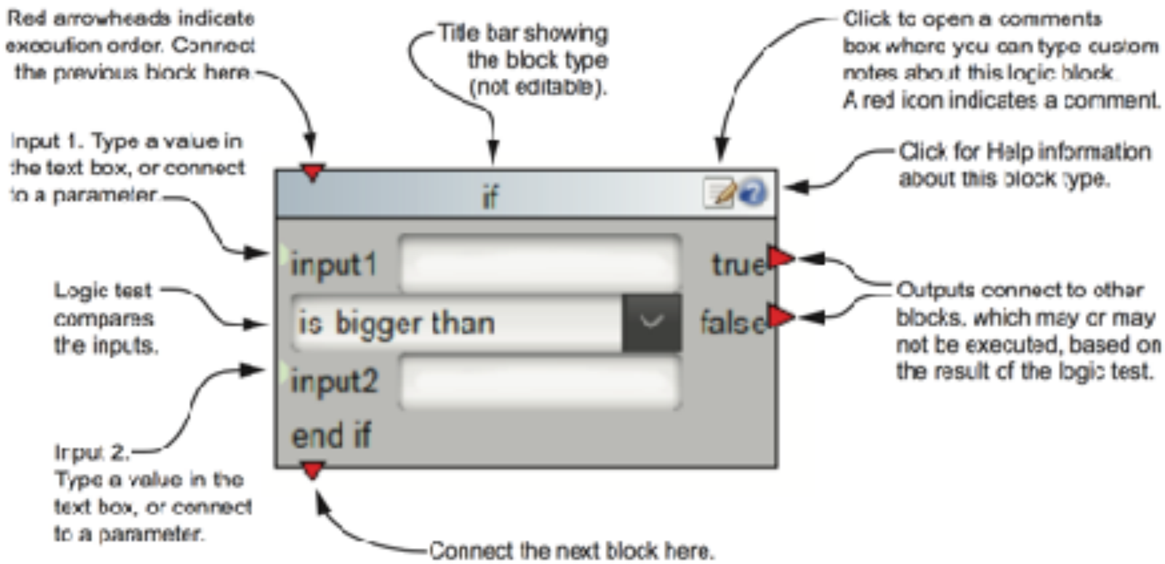


*Image 5 – The **If** Logic Block*

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.

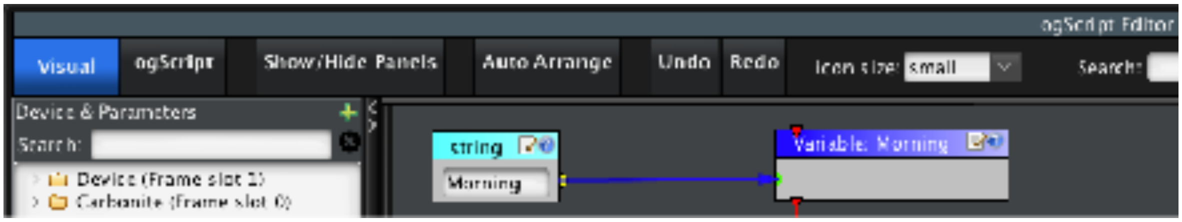


Image 6 – DashBoard v8 Visual Logic Interface Control