Version 8.7.1
Build Date: Monday October 7th, 2019.
This section describes new and improved functionality included in this release.

Memory Manager Indicator

The Memory Manager (Figure 1.1) displays the current memory usage of the DashBoard instance that you are running and when memory is low it takes actions to free up memory by unloading inactive tabs, as shown in Figure 1.2. Unloaded tabs are indicated by a caret symbol in front of the name, for example “^ CustomPanel01.grid”. The Memory Manager will not unload active CustomPanels or active tabs. If you have a panel that runs tasks in the background (listeners, GPI triggers, timers, and etc.), you may not want DashBoard to unload your panel. You can use the keepalive flag in the top level of the CustomPanel source code to indicate that this panel should not be unloaded. Existing CustomPanels will not be unloaded because the keepalive is enabled by default.

Note: The memory usage shown is approximate and subject to Java’s garbage collection schedule, and it may take a few moments for changes in memory consumption to be reflected in the status.

![Figure 1.1 Memory Manager with a healthy status (green).](image1)

![Figure 1.2 Two frames that are shown in unloaded state.](image2)

If your status indicator is green, then the memory usage is within the acceptable range. The status indicator turns yellow to indicate caution, and finally escalates to red to indicate when memory usage exceeds the recommended levels.

You can disable or enable the unload feature in the DashBoard General Preferences or set your preferences at the CustomPanel level.
Memory Manager Tag

If you would like to see a memory manager indicator directly in your panel, you can add the new memory manager widget. You can also customize its size and position when you add the memory manager tag.

In PanelBuilder Edit Mode, you can double-click on the canvas to open the Component Editor. From there you can click Source tab, and add the memory manager widget to the top level of the source code (usually an <abs> container). You can include attributes to modify the size and position of the memory manager status bar, as shown in the example below:

```
<abs contexttype="opengear" id="_top" keepalive="false" style="">
  <memory height="50" left="1500" top="50" width="200"/>
</abs>
```

Keep Alive tag

If you have a CustomPanel that should never be unloaded, you can set a Keep Alive flag in the Abs Attributes that will tell DashBoard not to unload this panel (even if the memory is low).

**Note:** CustomPanels that were created in DashBoard 8.6 and earlier will not be unloaded when memory is low, because by default the Keep Alive option is enabled on CustomPanels that were built before the Memory Manager was available.

In PanelBuilder Edit Mode, double-click on an empty area on the CustomPanel to open the Component Editor. The uppermost abs should be selected in the tree. In the Abs Attributes tab under Remote Task Triggering, select Keep Alive. This button prevents DashBoard from unloading this CustomPanel, even when memory is low.

You can see the Keep Alive button below:
You can also set the keepalive tag in the source code in the top-level abs. For example:

```xml
<abs contexttype="opengear" id="_top" keepalive="true" style="">
...main panel content here...
</abs>
```

**Ulritouch 4 Support**

You can now use the Ulritouch 4 Panel template when creating a CustomPanel for the new Ulritouch 4. The template provides a 1304 x 485 pixel canvas, which is designed to fit beside the built in DashBoard navigation in the Ulritouch User Interface (UI).

![Ulritouch 4 Panel Template](image)

**Color Chooser** now supports RGBA output format

You can now use the `w.format` config option to modify the Color Chooser widget. This option allows the user to set the output format for color strings to RGBA (red-green-blue-alpha) or ARGB (alpha-red-green-blue). The value can be set to `rgba` or `argb`, where `argb` is the default output format.

![Color Chooser Configuration](image)

**Tip:** You can find a list of the available config options in the Param Attributes tab > Config Options area by clicking the Help button.
Support for OGP Minimal Mode

DashBoard 8.7.1 now supports any devices that implement OGP Minimal Mode to request a minimal amount of status updates to DashBoard until the devices’ user interfaces open.

Additionally, the DashBoard Proxy server enables OGP Minimal Mode support by default for any devices it shares.

New DashBoard Logo

DashBoard’s User Interface now features the new Dashboard logo and updated style.

Other Enhancements

- Improvements to the **Color Chooser** widget which now supports the use of transparency with the ogScript `setStyle()` function.
- Improvements to Proxy Server support for Carbonite.

Camera Enhancements

Camera enhancements include the following:

- Support for Multi-Camera Mode for Sony Cameras Connected Through a CNA-1 Adapter
- **Sony Master Control**
- **PTZ Camera Control (for Sony BRC cameras)**
- Control of Sony BRC Cameras – VISCA Gateway
Support for Multi-Camera Mode for Sony Cameras Connected Through a CNA-1 Adapter

Ross Video’s Sony Paint plugin for DashBoard now supports Sony’s HZC-MSCN1 software upgrade for CNA-1 units. The software upgrade, available from Sony, enables you to control multiple cameras from a single CNA-1 unit.

When you add a supported Sony ENG/Box Camera in DashBoard, you specify one of the following two operation modes:

- **RCP Mode** — Each camera is associated with a dedicated CNA-1 Adapter. The camera control computer(s) address each camera directly through the camera’s CNA-1 Adapter. RCP mode is sometimes referred to as Gateway mode.

- **MSU Mode** — The system includes one or more CNA-1 adapters, upgraded to run in MSU mode (Sony upgrade HZC-MSCN1). Each CNA-1 adapter controls a group of cameras. In facilities that have a large number of cameras, deploying multiple CNA-1 adapters may improve response time, as the system queues control requests and polls the cameras for status updates. An additional Sony MSU or upgraded CNA-1 adapter is required to act as the master device on the Sony network.

If you already have cameras that are configured for previous versions of the Sony Paint plugin that are connected directly (as opposed to via proxy server), they will continue working normally after you install DashBoard v8.7.1. Cameras connected via proxy server may need to be Unshared and then Shared to re-establish connectivity. See the Known Issues section of this document for more information (CI-190).

**Note:** The Sony Paint plugin is a licensed feature of DashBoard. It must be purchased from Ross Video. For assistance with licensing, please contact Ross Video Technical Support and ask for the Sony SSP Camera Control feature.

For detailed installation and configuration instructions, see the manual, “Installation Guide for Sony Paint Control (8351DR-012-8.7)”.

**Sony Master Control**

In this release, the Sony Paint plugin also includes a new configuration page that enables you to view and manage your connected cameras in both RCP and MSU mode. This Master Control page can be accessed from Slot 0 of the Sony ENG/Box Cameras node in the DashBoard tree:
PTZ Camera Control (for Sony BRC cameras)

DashBoard can now be used to configure and control select Sony BRC cameras models. DashBoard’s PTZ Camera Control plugin communicates with Sony cameras using VISCA protocol over IP.

DashBoard currently supports Sony models BRC-X1000 and BRC-H900.

Using the PTZ Camera Control plugin, you can:
• Store, recall, and manage presets
• Control camera pan, tilt, zoom, focus, and iris. These can also be controlled using an external joystick connected to the DashBoard computer.
• Perform basic paint and shading.
• View and configure camera settings, and access the camera’s on-screen configuration menu.
• Export configuration and presets to be imported later, for use with the same model of camera.

For detailed information about how to access and use the PTZ Camera Control plugin, see the Camera Control Manual (8351DR-015-8.7).

Note: The PTZ Camera Control plugin for configuration and control of select Sony BRC cameras is a licensed feature of DashBoard. It must be purchased from Ross Video. For assistance with licensing, please contact Ross Video Technical Support and ask for the Sony BRC Camera Control feature.

Control of Sony BRC Cameras – VISCA Gateway
The PTZ Camera Control plugin includes a gateway that enables controllers such as Carbonite to connect to supported Sony BRC cameras using VISCA Protocol over TCP.

Once you have added a Sony BRC camera to DashBoard, you can enable the VISCA gateway by navigating to Config > Gateway, specifying a port for your VISCA gateway, and tapping Enable to connect. The VISCA Gateway control page displays the status of the gateway and how many controllers are connected to it.

The VISCA Gateway provides access to:
• Directly control pan, tilt, zoom, focus, iris, and color on the connected camera.
• Store and recall presets. You can select whether to store presets:
  o Directly on the camera (set **Presets** option to **Camera**).
  o Locally on the DashBoard computer, where they can also be accessed through the **Presets** panel (set **Presets** option to **Local**).

**Note:**
• To access the VISCA Gateway feature for a Sony BRC camera, you must purchase a license for the **Sony BRC Camera Control** feature. Please contact Ross Technical Support for assistance.

**Bug Fixes**
The following bugs have been fixed in 8.7.1:
- You cannot use both the VDCP task server port and HTTP trigger server port at the same time for GPI triggers. (Navigate to **Component Editor > Remote Task Triggering**)
- The Color Chooser transparency slider does not show transparency.
- Issues with Proxy Server for Carbonite.
- Long OGP JSON messages block communications.

**Known Issues**
The following known issues have been identified in 8.7.1:

**DashBoard**
- If you use a panel that uses RossTalkEx to send commands to an XPression, and then try to access that XPression's Sequencer View panel, it will fail to load the sequence items. If you open the Sequencer View panel first, and then use your panel you will not encounter any issues.

**Paint Control for Sony ENG/Box Cameras**
- Upgrading to a new version of DashBoard causes SmartShell to lose connectivity to Sony CNA-1 cameras via the DashBoard proxy server. To re-establish connectivity, open the DashBoard Proxy Server page, make note of the order of the cameras, Unshare all cameras, then Share each camera again in the same order that they were originally shared by right-clicking on each camera in the DashBoard tree and clicking **Share Device**. (CI-190)

**PTZ Camera Control (for Sony BRC Cameras)**
- Focus control does not work from Carbonite (when configured to connect to a Sony device) or Acuity switchers when connecting to a PTZ camera through the DashBoard VISCA gateway. Auto focus is unaffected by this issue and can be set from Carbonite or Acuity. For Carbonite, the recommended workaround is to use the “pivotcam” device configuration when connecting to the DashBoard VISCA gateway. For Acuity, focus must be controlled directly from DashBoard. (CI-189)
- The full range of iris values is not available when controlling iris from Carbonite and Acuity when connecting to a Sony BRC camera through the DashBoard VISCA gateway. This does not affect the range of iris values when the camera is in automatic exposure mode. Workaround is to control iris directly from the DashBoard UI. (CI-188)
- When connecting to a Sony BRC-X1000 camera from DashBoard and switching from manual exposure to auto exposure with no change in exposure settings (iris, shutter, gain) there is a small but noticeable change in light levels in the video SDI output. This appears to be caused by the camera itself. (CI-169).
Version 8.6
Build Date: May 1, 2018.
This section describes new and improved functionality included in this release.

Wizard Feature
You can create wizards that contain a title, a page navigation pane, and a progress bar. The wizard allows you to automate complex tasks and break them into a series of steps that walk users through the process from start to finish. You can see an example wizard below:

![Wizard Example](image)

You can choose which features you would like to be visible, and how many pages appear in the wizard. Feature options include the tabs, progress bar, and dialog width and height. The wizard is now available in the PanelBuilder **Edit Mode** toolbar.

![PanelBuilder Edit Mode](image)

If you would like to modify the basic wizard, you can customize it using the script functions defined below. Each function is optional, so if you do not define a function, the wizard simply uses the default wizard settings for that function call. This allows a developer to override only a
small portion of the wizard’s operation, or to control as many aspects of the wizard’s operation as needed. You can see a basic example below:

```xml
<wizard dialog="false" height="465" left="104" name="My Wizard" style="bdr:shadow;" tabsvisible="true" top="122" width="694">
  <config help="" helptitle="" key="w.model" message="">var model = {
    getPageTitle: function(page)
    {
      return "SCRIPTABLE PAGE: " + (page + 1);
    },
    getMessage: function(page)
    {
      return "My message for page " + page;
    },
    getHelp: function(page)
    {
      return "My help for page " + page;
    }
  };
  model</config>
  <abs id="my-page1" name="Page One"/>
  <abs id="my-page2" name="Page Two"/>
  <abs id="my-page3" name="Page Three"/>
  <abs id="my-page4" name="Page Four"/>
</wizard>
```

Help Control

You can create a Help Control popup to display a custom title and help message using the Help tag, as shown in the example below.

The Help Control pop-up is now available in Panelbuilder’s Edit Mode menu.
To create a help pop up, use the Help button to create a basic canvas container, and in the Tag Attributes you can set the width, height, title and message. Messages can be plain text or html with many common html tags supported including hyperlinks. An OGLML example is shown below:

```html
<help height="40" html="true" left="53" name="?"
popupheight="200" popupwidth="500" style="bg#ff0000;"
title="Example Help" top="315"
width="40"> <![CDATA[<html><left><b><u>Html formatted Heading</u></b><br><font color=#ffffdd>Take me home</font><br><a href="https://www.rossvideo.com/">Ross Video</a><br>The latest software release for Carbonite Black Solo unlocks a powerful USB Media Player functionality and is available to customers at no additional cost. This new media player provides the functionality of a single-channel clip player, for playout of compressed MPEG-4 AVC media directly from a connected USB-media drive. There is no other production switcher in the world with this level of built-in media playback. ]]></help>
```

Ultritouch Delete feature

Ultritouch now provides a Delete feature that allows you to delete folders and CustomPanels from the Ultritouch User Interface. You can delete folders or CustomPanels from the Ultritouch device in DashBoard, or from the Ultritouch device interface. In DashBoard open the Ultritouch device, and navigate to Manage CustomPanels. Then select the folder or file you would like to delete, and click Remove, as shown below.
You can also delete CustomPanels in Ultritouch from the device interface using the same method, as shown below.

**Border Layout Container**

You can use the border layout tool to create an area on a CustomPanel that you can anchor components to and later resize to maintain your intended layout. You can use a border layout to anchor components against any of the four borders of the container and in the center and set which of these areas will maintain their proportions when resized. It is useful for adding menus along the border edge of a CustomPanel, or to group components within a CustomPanel and later resize the components easily.

To add a border layout, in the PanelBuilder Edit Mode toolbar navigate to **Grids, Tables > Border Layout**. Once you’ve drawn the border layout area on the canvas and set a growth quadrant (the only component that will expand when resized), you can add up to five components to the border layout. The components that are not part of the growth quadrant will maintain their height or width when resized.

Here’s an example of a border layout with the growth quadrant set to the central content:
The source code for the border layout example is:

```xml
<abs contexttype="opengear" gridsize="20" id="top" style="">
    
    <borderlayout height="460" left="20" top="40" width="780">
        <label anchor="north" height="56" name="Top Content" style="txt-align:center;bg#8C43AC;size:Big;font:bold;" width="182"/>
        <label anchor="east" height="370" name="Right Content" style="txt-align:center;bg#E67E22;size:Big;font:bold;" width="150"/>
        <label anchor="south" height="100" name="Bottom Content" style="txt-align:center;bg#287EB8;size:Big;font:bold;" width="467"/>
        <label anchor="west" height="338" name="Left Content" style="txt-align:center;bg#BF3A2B;size:Big;font:bold;" width="150"/>
        <label anchor="center" height="127" name="Center Content" style="txt-align:center;bg#27AE5F;font:bold;size:Big;" width="152"/>
    </borderlayout>
</abs>
```

Other Enhancements

- The option to set the visibility of components in the UI is now available for all component tags. The **Visible** checkbox, as shown below, is found by double-clicking a CustomPanel component in **Edit Component > Position/Stretch Attributes** tab.
- Visual Logic blocks now available for new Carbonite RossTalk Commands.
- The exit tag, <exit/>, now allows you to open a new tab when closing the current tab.
- Drawers now have improved attribute functionality, which includes the following fields: name, ID, width, height.
- Global labels with a *.lbl extension can now be loaded using File > Open.
- Additional connection options were added to Ultritouch for IPS types, such as NK-NET, and RCP.

Bug Fixes

The following bugs have been fixed in 8.6:
- Ultrix sometimes disconnects and return with “Max connections exceeded” message.
- Ultrix reloads information in DashBoard too frequently in certain network environments.
- When a drawer is set to Enabled, it doesn’t hide or show the Drawer tab.
- DashBoard may run out of memory when loading very large firmware files.
- Issues with platform upgrades in Ultritouch.
- Unable to use multiple OGP devices on same IP with DashBoard Command Line Interface.
- The File Browser does not point to the real location when the browser file is clicked.
- Dragging params from Carbonite to the CustomPanel breaks VL API blocks.
- The exit tag’s Exit Type Window opens more than one exit box.
- DashBoard cannot send firmware to NK products.
- For the XPression Plugin, you cannot see the Pattern Generator in DashBoard unless DashBoard is local to XPression.
- For the XPression Plugin, adding a device in config makes the XP widget not show the take items for all the XP devices.

Known Issues

The following known issues have been identified in 8.6:
- N/A

Version 8.5.1

Build Date: December 3, 2018.
This section describes new and improved functionality included in this release.

Drawers

If space is limited on your custom panel, you can now create drawers to make additional space for content. This is ideal for smaller panels with restricted space, such as the Ultritouch custom panel, or any panel that is crowded with too many components. It can help to organize your content, compartmentalize standalone functions, or to minimize certain parts of the custom panel when it is not in use. You can see an example below:
Drawers are now available in PanelBuilder’s **Edit Mode** menu, and have been consolidated under the previous **Tab & Split** button, which has been renamed to **Tab, Split & Drawer**. To create a drawer using PanelBuilder, you must first create a basic canvas container, a drawer component area, and then add individual drawer tabs to any side. To affix a drawer to a side, you must set the drawer tab **Anchor** attribute to **North**, **East**, **South**, or **West**. Note: when you add a basic canvas to the drawer component area it will automatically appear as a drawer. Optional tab styling options are available.

For more information see, the *DashBoard User Guide*.

**Pager Control**

If space is limited on your custom panel, you can now create a series of pages for your content and a pager control to enable users to access other pages. This is ideal for smaller panels with restricted space, such as the Ultritouch custom panel, or any panel that is crowded with too many components. You can use Pager controls in conjunction with drawers to get the most out of your space. You can see an example below:
For more information see, the DashBoard User Guide or the CustomPanel Development Guide.

Other Enhancements
- You can now enable or disable continuous layout in `<split/>` tag with the following:
  continuouslayout=true/false.

Bug Fixes
The following bugs have been fixed in 8.5.1:
- The NDI 3.7 issue that previously required Visual C++ Redistributable Packages for Visual Studio 2013 was resolved.
- The issue with sending legacy RossTalk to commands to XPression using RossTalkEx in Visual Logic is now resolved.
- Issues with the Ultritouch UI were resolved:
  o Ultritouch loading screen no longer takes up the entire screen.
  o Ultritouch loading screen text, “Waiting to load”, can now be read over any background color.
- Issues with the File browser were resolved:
  o In some scenarios, the file browser dialog box may display in a small window. This can be resolved by enlarging the parent window before opening the dialog box.
  o Images load slowly on certain systems.
- The foreground cannot be set for the combo box parameter, and it will show an error message.
- The `<api/>` continues processes even when its’ owner context is closed.
- Drag-to-pan appears inconsistent in `<page/>` tag.
- Drawer
  o Issues with the drawer reloading while using PanelBuilder in Edit Mode were resolved.
- Issues with DashBoard being unable to recover multiple times from a corrupt CustomPanel OGD file were resolved. Note: previously DashBoard would recover the first time, but not afterwards unless the *.bad file was deleted.
- The issue with multiline text fields was resolved, and has been updated to show read/write vs. read-only state.
Known Issues

The following known issues have been identified in 8.5.1:

Version 8.5
Build Date: December 14, 2018.
This section describes new and improved functionality included in this release.

Ultritouch CustomPanel Template
You can now create custom panels for Ultritouch, an adaptable system control panel with monitoring capabilities.

To create an Ultritouch custom panel:
2. Set Folder to the folder of your choice.
3. Enter a file name.
4. Set Template to Ultritouch Panel and click Finish.
PanelBuilder displays the template’s canvas layout of 1304 x 203 pixels, to match the dimensions of the Ultritouch display screen.

Set the Visibility of Components in an <abs/> Tag
The “visible” flag allows users to set the default visibility of components in an <abs/> tag.

RossTalkEx Command added to the XPression Plugin
Dashboard and XPression now support a new protocol for sending commands, called RossTalkEx. Commands to send messages using that protocol are now available as part of the XPression Plugin. Users who have an XPression device can use RossTalkEx commands to trigger specific actions. These include setting a datalink key or setting template data. RossTalkEx is also available as an ogScript command.
To call a general-purpose function, use:
rostalkex.function name(parameters);

For example:
rostalkex.sendMessage("10.3.2.1", 8020, "DATALINQKEY 101:k1:v1", null);

Grid Creation Tool Added to XPression Devices
A “Pattern Generator” panel has been added to XPression Devices.

When you open that panel, you will see something like this:

This tool can be used to create “grid” images for modular displays (like in stadiums). Those images can be displayed on those screens using XPression. They can be used to verify that the screens are properly wired, and do not have any defective pixels.

Other Enhancements
- Updated to NDI 3.7.
- Improved NDI frame rate when running in “high quality” mode.
- SLP automatic device detection is disabled by default and can now be enabled in Window > Preferences.
- Added support for relative URLs from DashBoard Connect devices.
- Add padding to table parameters with \texttt{w.padding=top,left,bottom,right}.
- Add support for \texttt{w.rowheight} to list parameter.
- Add support for OGLML styles inside of combo box parameters.
- Add ability to disable tick marks on sliders with \texttt{w.noticks=true}.
- Add ability to jump directly to mouse on slider when clicked with \texttt{w.jumptoclick=true}.
- Added ability to load Global Labels file from File > Open menu.
- Added test-pattern generator to Xpression plug-in.
- Added the ability to access the countdown field of XPression take items using the XPression Plug-in.
- Added the ability to send RossTalkEx commands to XPression. There is a new “rosstalkex” object in ogScript to send those types of command. RossTalkEx is a new secure protocol modeled on RossTalk.
- Added the "set datalinq key" and "set template data" visual logic blocks for XPression Devices (these only work with RossTalkEx).

\textbf{Bug Fixes}

The following bugs have been fixed in 8.5.0:
- Devices with an icon, such as Graphite or Carbonite, with a frequently changing status no longer cause DashBoard to run out of memory.
- An issue that could cause DashBoard to hang when resizing OGLML user interface with embedded web browsers was resolved.
- The issue with saved frame configuration files that did not include cached external objects and HTTP content was fixed.
- The issue with configuration properties being removed from parameters in the Param Editor dialogue has been fixed.

\textbf{Known Issues}

The following known issues have been identified in 8.5.0:
- Sending legacy RossTalk to commands to XPression using RossTalkEx does not work in Visual Logic.
- Ultritouch
  - Ultritouch loading screen takes up entire screen, and requires UI adjustments.
  - Ultritouch loading screen text, “Waiting to load”, cannot be read with certain background colours.
- File browser
  - In some scenarios, the file browser dialog box may display in a small window. This can be resolved by enlarging the parent window before opening the dialog box.
  - Images load slowly on certain systems.
- The foreground cannot be set for the combo box parameter, and it will show an error message.
- The \texttt{<api/>} continues processes even when its’ owner context is closed.
- Drag-to-pan appears inconsistent in \texttt{<page/>} tag.
Version 8.4.1
Build Date: June 19, 2018
This section describes new and improved functionality included in this release.

URLs Supported in <ext/> Tags
CustomPanels can now use URLs as the target for the <ext/> (Panel Link) tag. Clicking on one of these tags will open a browser window pointed to the specified URL.

Other Enhancements
- File browser now displays image thumbnails.

Bug Fixes
- An issue when creating multiple buttons using Visual Logic in a CustomPanel to control an Ultrix Router was addressed.
- An issue with text search in the ogScript editor occasionally skipping-over matches was addressed.
- During device initialization, if a parameter or external object change trap was received by DashBoard, the original descriptor may have been loaded instead of the updated data.

Version 8.4
Build Date: June 6, 2018
This section describes new and improved functionality included in this release.

XPression Plugin
The XPression plugin allows users to add an XPression device as a node in the DashBoard tree. This node provides access to an XPression panel in DashBoard, from which users can remotely control and monitor the XPression sequencer. With this panel, users can create take items, change their properties (including published fields), and control their on-air state. Users can also create custom panels that use any of the elements of the XPression panel.
To use the XPression plugin, you must have an XPression system running XPression version 8.0 (or higher). DashBoard must have access to port 8020 on the XPression server.
A license is required in XPression to use the plugin in DashBoard. Contact Ross Video Sales or Technical Support for more information.

Support MIDI Panel Input
Similar to joystick support in DashBoard 8.3, DashBoard 8.4 allows external MIDI control panels to be attached to parameters in the Graphite audio mixer, paint parameters for ACID cameras, or arbitrary parameters in Custom Panels.
ogScript Debug
There are two new buttons when viewing the creation of a task in DashBoard. Both the Debug and Run buttons are in the top right corner of the ogScript Content area of where a task is entered, whether in Visual Logic or ogScript mode.

Run
A very useful feature for DashBoard where you can now execute the task as it has been entered allowing you to ensure that this specific portion is correct. This is a great tool for debugging!
Debug

When clicked, the following **Attention** dialog box appears.

Selecting **Yes** opens an ogScript Debugger:
Visual Logic Additions

- Support for Graphite
  The Graphite API is the same as a Carbonite API, but it may be expanded in the future.

- Improved Ultrix Performance
  DashBoard v8.4 includes numerous enhancements that significantly improve performance when interacting with Ultrix systems. Ross Video recommends that Ultrix users upgrade to DashBoard v8.4 as soon as possible.

Other Enhancements

- Decrease memory footprint of large devices inside of DashBoard
- Latest bug fixes for Visual Logic and other features
  - Allow existing parameters to be selected for visual logic devices
  - Allow setting of parameters from different contexts
- Enable shut-down of DashBoard via command line
  DashBoard.exe --launcher.openFile shutdown
- New and improved file picker
- Add style parameter widget hint for String parameters ("style")
- Add onresize ogScript event
- Performance enhancements to start up time
- Save DashBoard Basic Tree connections as they change instead of only on exit
- JSON improvements
  - Change JSON protocol logic to use a non-blocking way connection mechanism. Should greatly-increase performance when connecting to JSON devices when one or more is unreachable.
  - Fix bug where CPU would spin at 100% when a JSON device was connected
- Commands
  - Added support for Commands in Visual Logic
  - Support dropdown lists in Commands in Visual Logic
  - Fix issue with adding a Command in Visual Logic from a freshly-added device
• Added `<borderlayout/>` tag with support for components anchored "north" "east" "south" "west" and "center"
  It is a modified layout that also allows you to specify one of the quadrants to grow as large as it can without postage stamping
• Allow Linux to use `<webget/>` instead of browser
• Update OSGi core plug-in to add aliases for Windows Server 2012, 2015, and 2016 so binary contributions work
• Add a touch-friendly version of OGP Tree Widget (w.touchtree=true config property)
  1. Create a Tree Parameter with a widget hint of “Tree”
     Use +”Item” for top level and additional + added for sub-layer
  2. Add the parameter onto the screen.
3. Edit the parameter and add a “w.touchtree” with a value of true to the Config Options.

- Add ability to specify styles for text field and button for file picker OGP param widget:
  1. Create a Parameter with a File Picker Constraint Value Widget Hint.
  2. Add the Parameter onto the screen.
  3. Add w.buttonstyle with a value in the Config Options: area to define the button style.
  4. Add w.labelstyle with a value in the Config Options: area to define the label style.

- Allow JSON Proxy Server to require password authentication with -share-password PASSWORD launch argument when someone tries to open a shared device in the DashBoard Proxy Server:
  1. Add a new TCP/IP Device as a JSON device
  2. Option #1 – Edit the DashBoard shortcut:
     a. Right click on the DashBoard shortcut and Properties
     b. Edit the target to include -share-password “PASSWORD” at the end of the string (replacing “PASSWORD” with the desired password
     c. The example below sets the password to Greg:

3. Option #2 – Edit the DashBoard ini file:
   a. Locate the DashBoard.ini file on the computer and open it in an editing tool
   b. Add in -share-password on one line and the desired password on the next line
   c. The example below sets the password to Greg:
• Support onmouseenter and onmouseexit mouse events:
  1. Add a button with a defined Name ID:

  ![Button Attributes](image1)

  Name: TAKE
  ID: TakeButton
  Type: Push

  2. Scroll to the top of the source code and add in a new ogScript tag.
     a. Change the Event Type(s): to be either onmouseenter or onmouseexit.

  ![Event Type(s):](image2)

     Target: onmouseenter

  b. Add a task that will be executed when the mouse condition occurs.

     When the mouse goes onto (or leaves) the button it will execute the defined task.

• Add "speed" control to joystick axis controls.

  ![Axis Controls](image3)

  After adding a USB joystick and double clicking on the added joystick, the Axis Controls section includes a new column for Speed %.

  ![Axis Controls](image4)
Bug Fixes

- Fix file watcher in file navigator to use a single instance/thread
- Fix bug where each task had its own RossTalk object and was unable to share an active connection
- Fix bug in table widget when editing unconstrain float32 (wrong precision shown)
- Fix bug in visual logic that didn't show parameter OIDs from "legacy" visual logic custom panels
- Fix bug that prevented toggle button type from working properly on joysticks
- Fix bug in Visual Logic caused some functions with return values to be treated as though they don't (didn't show the output connector in Visual Logic)
- Fix bug that prevented toggle button, check box, and combo box values from being correctly-restored when using String/String choice constraints where the value is different from the display name
- Fix bug that prevented DashBoard from loading images from HTTP servers with multiple Cache-Control header values
- Fix bug that prevented DashBoard from correctly-loading content from web servers using non-ASCII Unicode characters

Version 8.3.1
Build Date: July 24, 2017
This section describes new and improved functionality included in this release.

Bug Fixes

- Fixed a bug to ensure listeners are always called when updates to individual array values happen in rapid succession
- Visual Logic setParameter now lets you choose the param from a drop-down list
- Revised exception thrown while processing a panel that could show undesired information
- If there is an exception when reading a parameter XML file, OGD file, or ogJSON file DashBoard will look for the most recent backup and, if it exists, will copy it over to replace the corrupt file. Corrupt file will be renamed to *.bad and an entry will be added to the log.

Version 8.3
Build Date: June 27, 2017
This section describes new and improved functionality included in this release.

NDI™

NDI™ is now available in CustomPanels in the edit toolbar. Adding an NDI™ stream to DashBoard is now easy and straightforward.
The NDI™ plugin now does auto discovery of NDI™ streams and a slight reworking of the edit mode tag attribute UI:

Note that no video is displayed when running DashBoard on a Linux 32bit operating system.

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

Widget Config Options
Widgets have had additional configuration available by manually adding them to custom panels.
For most users, knowing these exist as well as knowing how to add them has made these features unavailable to most creators of CustomPanels.
The Param Attributes Editor now has an editor for all config options.
Example showing an instance of it for the ‘Date Picker’ widget:
The ‘?’ icon to the right of the ‘Config options’ table gives a popup that shows ALL of the options for each widget, a snippet of it is as follows:

Other Enhancements

- `ogscript.asyncFTPListFiles(host, port, username, password, path, callback)` can now be called with a file name
  - Updated method signature is `ogscript.asyncFTPListFiles(host, port, username, password, path, fileName, callback)`
  - File name must be within given path
  - File name may contain the * wildcard Ex: '*.grid'
- Fixed bugs in loading joystick mapping files designed for different joysticks
- Fix bug in Visual Logic block for setStyle

USB Joystick Support

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

To Add a Joystick to DashBoard:
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:
The joystick is active in this mode and you can move axis and press buttons to see which item is which.

Note that if the joystick is unplugged/replugged it can result in a loss of control. To return control, restart the application. There can be times where the joystick status does not get updated in the DashBoard tree, wrongly indicating that it is working/not working.

**ogScript Additions**

The following have been added to ogScript:

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

```javascript
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.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
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](image1)

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

[Ok] [Cancel]

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”.

![TestDevice (MyDevice)](image2)

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.
  
  **NOTE**: 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.
The following shows the 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 manually.

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

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:

```xml
<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.

Other 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
Release Date: September 1, 2016
This section describes new and 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
Release Date: July 11, 2016
This section describes new and improved functionality 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 CustomPanel.
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 Ultrachrome HR outputs for chroma key applications.

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 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).

The following shows the 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 Comtrol IP-to-Serial bridgedevice. 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.

The following shows the 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:

1. Install DashBoard v8.1
2. 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)
3. Right click on the frame you want to share and select “Share Frame”
4. In the DashBoard Services tree node, open the DashBoard Proxy Server (you should see your shared frame listed).

5. 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.

6. 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:

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

2. 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)
3. 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 8.0
Release Date: April 13, 2016
This section describes new and improved functionality included in this release.
DashBoard Version 8.0 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 8.0 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.

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.
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.

---

**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. The user can create additional APIs to control third-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.

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, tap 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.

Copyright Notice

© 2019 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.