

## CustomPanel Checklist

**Release Date: June 21, 2019**

This application note provides some tips and reminders to keep in mind when creating a DashBoard CustomPanel. While not required, these are recommended best practices.

### Before you begin

#### 1) Define the scope of the panel

While it may sound a bit silly the first thing that a DashBoard developer should do is figure out what the panel they are creating should accomplish. As DashBoard is an open platform, and is so powerful, projects can quickly expand to the point that a panel is never released. A defined scope, and ideally a quick sketch of the user interface, will help determine when the first “phase” is done, and then can be expanded on with future iterations.

#### 2) Determine all the devices you need to control

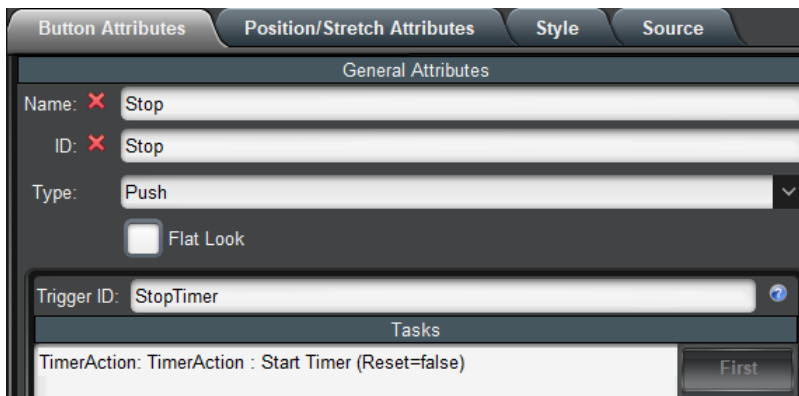
DashBoard is a perfect platform for controlling almost any product that Ross Video makes, and over 120 partners have integrated their products into the DashBoard ecosystem. DashBoard can send both TCP/IP and UDP messages, and it can also be used to control unsupported third party IP-based devices (or serial devices if you use a serial to IP converter). Listing the devices to control in DashBoard, and searching for their protocols, which can typically be found online, will help keep the CustomPanel development process rolling.

For devices that do not have APIs pre-built into DashBoard, they can be created as Visual Logic blocks to enable a user to call the function over and over without having to type in the code each time.

### Before you finish

#### 3) Ensure buttons/labels you wish to trigger remotely have a trigger ID

To access button remotely to perform a task the button must have a unique trigger ID. This can be entered by accessing Edit Mode (CTL + G on a PC or pressing the PanelBuilder Edit Mode button at the top menu of the DashBoard screen). Double click on an already created button to bring up the Edit Component <button> pop up. Under the Button Attributes tab on the right of the screen is a space for Trigger ID. This can be a string or a number (note that if you want to trigger a button from a Carbonite GPI or the Ross Video GPI card the Trigger ID must be a number).

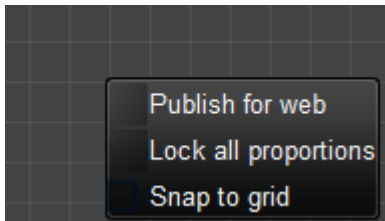


Showing the Trigger ID of StopTimer added to the button.

#### 4) Lock all proportions

This feature maintains the positioning of the panel even when the screen size has changed. To access this enter

PanelBuilder edit mode and right click on the custom panel anywhere there is no content. This opens a pop up that has three options (Publish for web, Lock all proportions and Snap to grid – Note: Snap to grid is a very useful option to turn on when starting to develop a custom panel to help line everything up nicely).



This is how the right-click pop up will appear.



This is how the right-click pop up will look appear.



Displays the original panel.



PIVOTCam IP Address: 10.20.35.49

Chain ID: 1

Shot Number: 36

Recall Shot

Same panel with DashBoard dimensions changed showing how an unlocked proportion panel cuts off user entry fields.



PIVOTCam I... [input field]

Chain ID: 1

Shot Number [input field]

Recall Shot

Same panel with DashBoard dimensions changed to show how a locked proportion panel maintains user entry fields.

### 5) Add “editlock” attribute and set “SimpleEncrypt”

There are two way to protect a panel from tampering. The first is to set a password so that if someone tries to enter PanelBuilder Edit Mode it will prompt them with a pop up asking for the password. This is set by entering the PanelBuilder Edit mode, and double clicking where there is no content to access the Edit Component <abs>, clicking on the Source tab on the right and editing the top line of the code to include editlock = “<password>” where <password> is what a user must type to enter edit mode in the future. In the code example below, you can see that the first line sets the panel password to RossVideo:

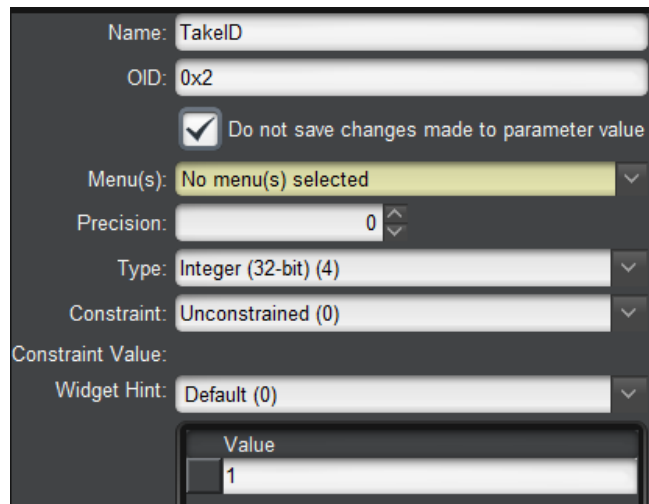
```
<abs contexttype="opengear" editlock="RossVideo" virtualheight="1077" virtualwidth="1918">
```

The second way to secure the panel is by encrypting the code. Setting a password a great best practice, but if someone can read the source code and see the password then it does not provide enough protection. To encrypt a panel so that the source code cannot be viewed, add encrypt="SimpleEncrypt" to the top level code. Using the above example and adding in SimpleEncrypt it would look like this:

```
<abs contexttype="opengear" editlock="RossVideo" encrypt="SimpleEncrypt" virtualheight="1077" virtualwidth="1918">
```

**6) Set stateless to true for any parameters whose value should not be written to the file**

To prevent a parameter from writing the value to the file, you need to set stateless to true. This is done by entering edit mode, clicking on the Parameters box in the edit mode window to appears when you enter edit mode. This opens an Add/Edit Parameters screen. Choose the parameter you want and it provides the information. Select the check box that says "Do not save changes made to parameter value". This will then add stateless="true" to the code for that parameter.



Add/Edit Parameters showing the check box to turn on stateless for a parameter

```
<param access="1" constrainttype="INT_NULL" name="TakeID" oid="0x2" precision="0" stateless="true" type="INT32" value="1" widget="default"/>
```

Sample code with stateless set to true.

**7) Set stream to false to stop the parameter from appearing in XPression DataLinq**

By default parameters created in DashBoard are accessible by XPression DataLinq. While this is quite useful, having every parameter made accessible creates an unwieldy list to choose from when setting a DataLinq value. A parameter can stop sending its information to DataLinq by setting stream="false".

```
<param access="1" constrainttype="INT_NULL" name="TakeID" oid="0x2" precision="0" stream="false" type="INT32" value="1" widget="default"/>
```

Sample code to stop sending parameter information to DataLinq.

**8) Disable output to the debug view**

Chances are the panel was created to send debug commands to test the panel. However, when the panel is being used these debug commands should be disabled. There are two ways to accomplish this, one that requires the user to set things up at the start of the panel.

The first option is to go to every instance of the ogscript.debug and comment them out. This can be time consuming depending on how many debug commands were added during the creation of the panel. It is also time consuming when wanting to debug the panel in the future as these commands would need to be un-commented to restore the

debug.

The second option is to create a "log" function in an `</api>` block at the start of the panel development process and to call that instead of the `ogscript.debug`. By doing this you can comment out the single line in the log function to turn off the commands, and restore it quickly for debugging purposes.

```
<api>
function logMessage(msg)
{
  ogscript.debug(msg); //COMMENT THIS LINE OUT TO TURN OFF DEBUG
}
</api>
```

#### 9) Provide a version number

While some panels may not be modified after they are created, it is never a bad idea of include a version number that is either visible in the user interface, or at a minimum, is included as a comment in the code. We typically use a version code of `major.minor.micro.yyymmdd` so that we can easily, and clearly, identify that the proper release is being used.

#### 10) Backup the panel

Once the panel is complete create a backup of the panel so that, if something happens, whether to the computer or DashBoard, there is a copy that can be restored. We store completed panels on an internal SharePoint server so that if a customer ever has an issue we can provide support or give them a backup version.

## Custom Panel Checklist

### Before you begin

- Define the scope of the panel
- Determine all the devices you need to control

### Before you finish

- Ensure buttons/labels you wish to trigger remotely have a trigger ID
- Lock all proportions
- Add “editlock” attribute and set “SimpleEncrypt”
- Set stateless to true added to any parameters whose value should not be written to the file
- Set stream to false to stop the parameter from appearing in XPression DataLink
- Disable output to the debug view
- Provide a version number
- Backup the panel