Axium Mobile Application Documentation

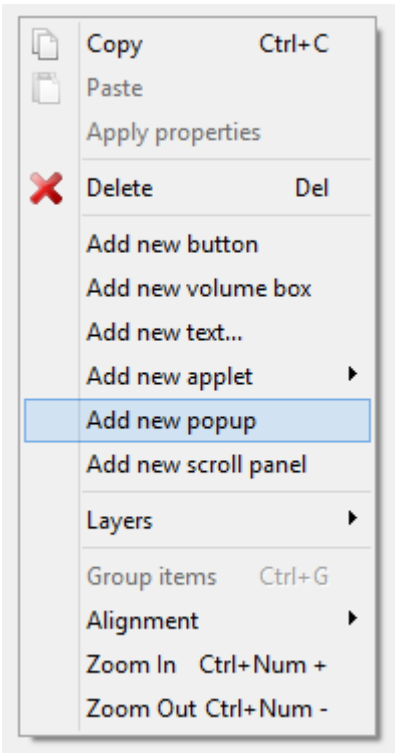
Most of this document is geared towards providing an overview on the usage of Axium Design Portal with regards to the Axium mobile app. However, the Common Issues section describes some issues that are possibly more relevant to the actual runtime of the mobile app itself.

Popups

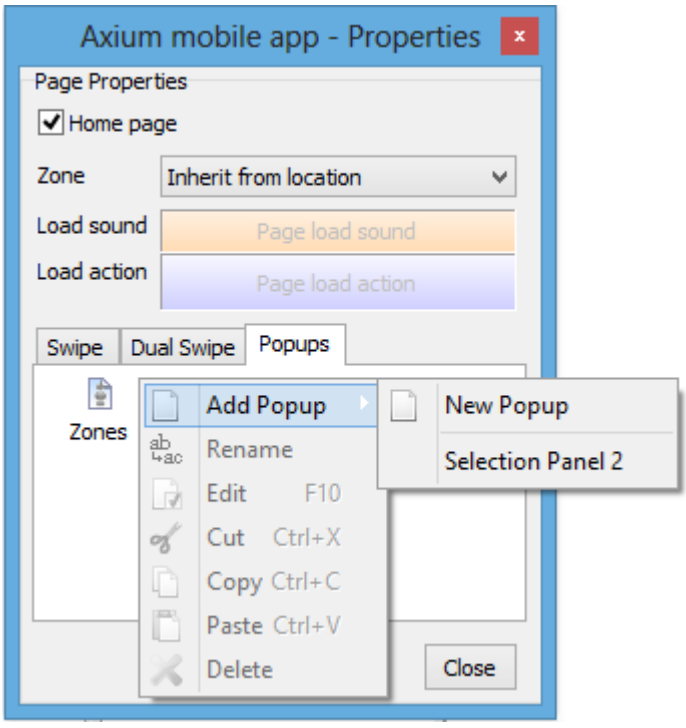
Popups are a feature that allows a frame of content to be shown and hidden by the user. They are useful for displaying confirmation or information dialogs and when displaying lists or selections that are too complex to be displayed as a permanent fixture on the page itself. Popups will often be used in combination with a Scroll Panel in order to display a large list.

To create a new Popup there are two common methods:

1. Right click the page designer and select ***Add new popup***
2. Right click the popups list in the page properties and select ***Add Popup.*** From here you can either create a new popup or add an existing popup to the page.



Method 1 – Right click page



Method 2 – Right click popup list

Once a popup has been created an empty frame will be added to the page. At this point the popup can be sized and positioned like any other page item. Popups are unique however, in that if another page item is selected, the popup frame on the page will become invisible. This is a designer feature

purposed so that the designer UI does not become cluttered with layers of page items. To select a hidden popup, go to the Popups list in the page properties.

To define the layout and contents of a popup, there are again, two methods:

1. Double click the visible popup on the page or popup list
2. Right click the popup in the popup list and select ***Edit***

The popup layout designer is essentially the same thing as the page designer. Current limitations are that a popup is unable to contain other popups and the swipe gesture actions are not yet implemented.

To close the popup layout editor and to return to the page designer, click the green tick in the top left corner of the designer toolbar.

Figure 3 – Finish editing popup

Another essential part of adding popups to the layout design is to set up the triggers to show and hide the popup. Commonly this is done as a button action but this can also be implemented as a part of a macro. There are a number of ways to set a popup as an action depending on the destination for the action.

1. Drag the popup from the popups list onto the desired button.
2. Right click an action box in the relevant properties dialog and select the popup from the ***Popups*** submenu.
3. Right click the desired button and select the desired popup from the ***Link To Popup…*** submenu.

Triggering the popup action will either show or hide the popup where appropriate. This allows a button to act a toggle for the popup.

Aside from adding existing popups as action there are also two other popup actions which can be set. These are ***Close All Popups*** and ***Close Popup***. The ***Close Popup*** action can only be set when the destination for the action is within a popup.

Scroll Panels

Scroll Panels are a useful feature which enables designing layouts to extend beyond the boundary of the containing layout. This is important and useful for displaying lists or a large number of related

page items which would ordinarily have to be spread across multiple pages in order to fit them in the design. They are also aesthetically pleasing and a natural input method for touchscreen devices.

To create a new Scroll Panel, right click the page designer and select ***Add new scroll panel***

The scroll panel layout editor follows the same principles as the page designer and the popup layout editor. The scroll panel has additional restrictions to prevent adding popups or other scroll panels to the layout of a scroll panel.

Scroll panels can be either vertically or horizontally orientated. This setting determines the axis of the scrolling content. The content size boxes in the scroll panel properties define the size of the scrollable area. One dimension will always be equal to the actual page item size while the other can be larger. A vertical scroll panel's content width for example must equal the page item width since it does not scroll on that axis. Its content height however can be any value larger than the page item width.

Paging is another useful feature of scroll panels. When this option is selected, the behaviour of the scroll panel changes so that the scroll is divided into segments or pages, each with the width or height of the page item depending upon the orientation. A non-paging scroll panel on the other hand is free flowing and is more suitable for displaying lists as opposed to layouts.

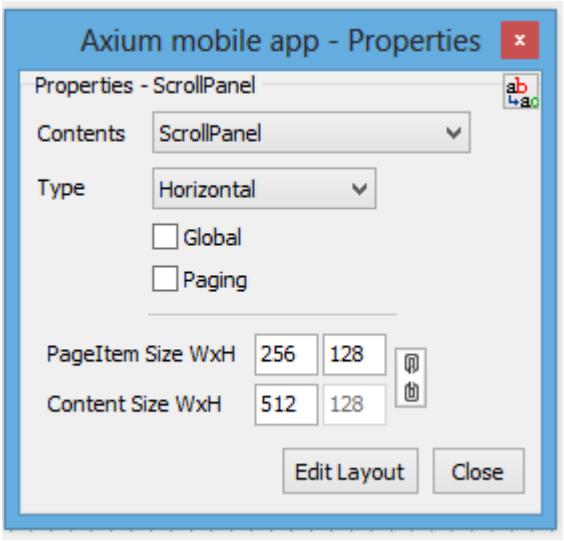


Figure 4 – Scroll panel properties

Landscape / Portrait Pages

Pages have three settings with regards to the page orientation: ***Portrait***, ***Landscape*** or ***Both***. Each orientation must be designed for individually and normally each orientations page layout would have comparable functions and items for consistency. Although the two orientations both logically represent the same page, they are in fact separate pages so rotating a device will load the other orientations page and clear any resources as needed from the last. This is important to keep in mind when using applets with regards to the concept of applet instances. This will be described more in the applets section.

The Portrait or Landscape pages can be added or removed at any time in the design process by checking or unchecking the appropriate checkbox in the mobile app design settings. Bear in mind that removing a set of pages is an operation that will not be able to be undone due to the magnitude of the changes required.

To change between the orientations in order to edit them, click on the rotate page button in the Resources toolbar.

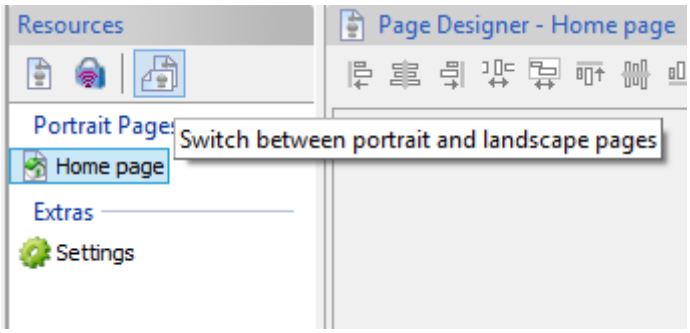


Figure 5 – Rotate page

Scaling / Resizing designs

Layouts designed with Axium Design Portal can to some extent, be scaled to suit devices with various screen sizes. There will inevitably be the need to manually position and resize certain items to suit different devices, especially when the aspect ratio differs greatly from the original and the scaled size. The primary advantage of scaling designs for supporting other devices is that all of the programming aspect is retained.

There are two options when resizing configurations, ***Scale*** and ***Move***. To access these options simply change the target resolution of an existing design.

Selecting the ***Scale*** resize type will adjust the size of all page items, respecting the aspect ratio, to the comparative size capable of fitting the new resolution. Note that the ***Font*** property on page items is not scaled to match the new resolution. The font size will need to be adjusted manually to suit the new size.

Selecting the ***Move*** resize type will simply change the resolution of the layout and any page items that are positioned outside the layout boundary will be moved in order to fit. This option may be a better choice if more fine-tuned handling is required for the resolution change.

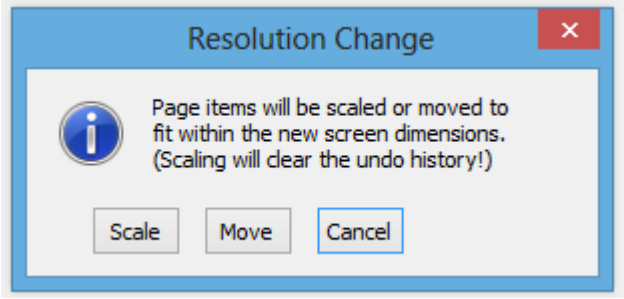


Figure 6 – Resize layout

Animations

Animations are an option that is available for actions which either transition pages or show and hide popups. They are set by performing a right click on the appropriate action in the relevant properties dialog and selecting an appropriate animation from the ***Transition Animation*** submenu. In the case of macros, the same procedure is followed but on the actual macro action listed in the macro instead.

The ***None***, ***Fade*** and ***Zoom*** options are self-explanatory, but the ***Slide*** animations have an alternate mode for each direction. When using a ***Slide*** animation the entry and exit behaviour of the views must be defined. This is presented in the animation selection menu under two headings: ***Slide Opposite Entry/Exit*** and ***Slide Same Entry/Exit***. When applying to a page transition, the entry and exit directions are applied to the entering and exiting pages respectively. When applying to a popup action, the entry direction is used for showing the popup, and the exit direction is used for hiding the popup.

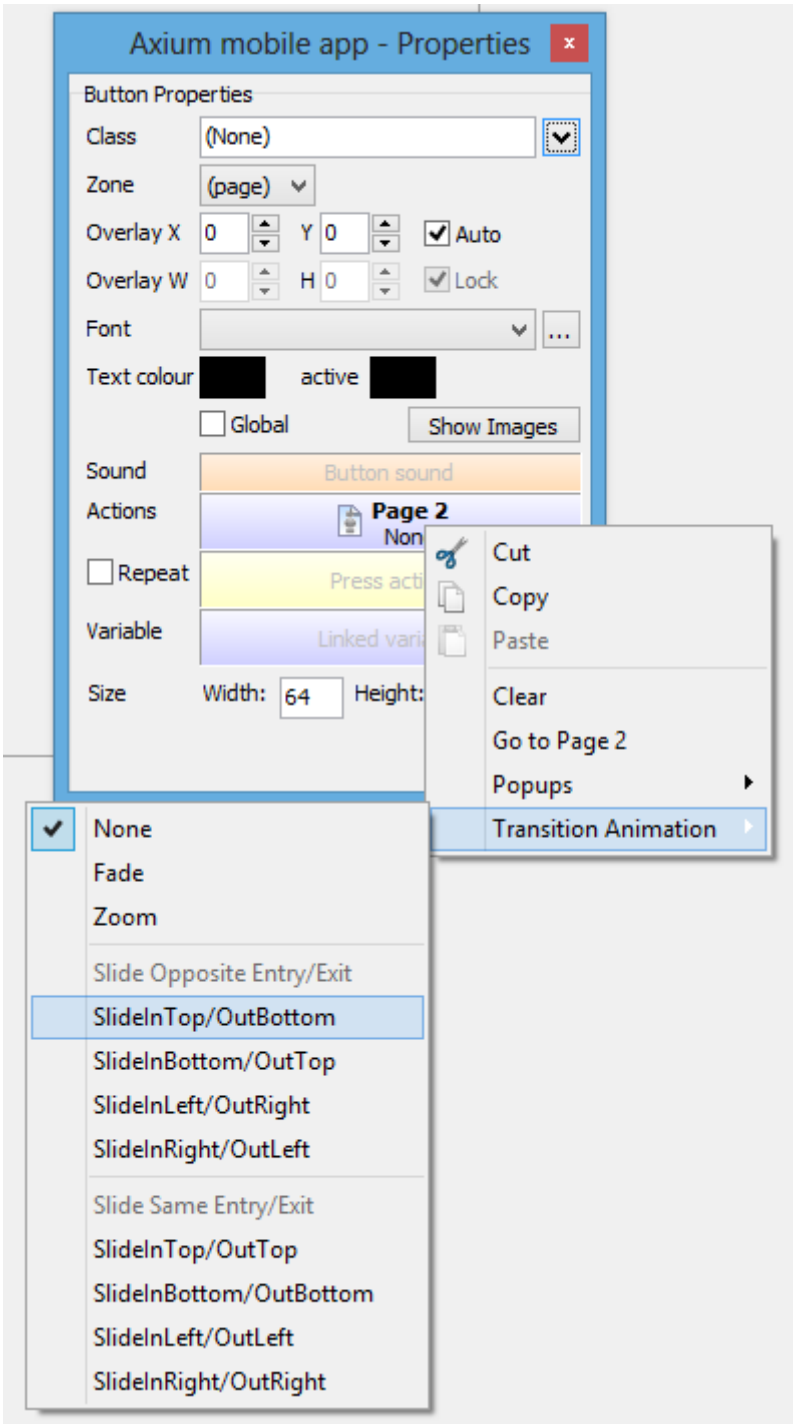


Figure 7 – Set an animation for a page transition

Variables

Leveraging the use of variables in the Axium mobile app is a feature made available when utilising an Axium Controller in the system. The usage possibilities for variables are vast and have applications for scripting, reporting sensor feedback, and for providing state information for devices which don't report it themselves.

To add variables, select the ***New Variable*** toolbar button when configuring an Axium Controller device in the project.

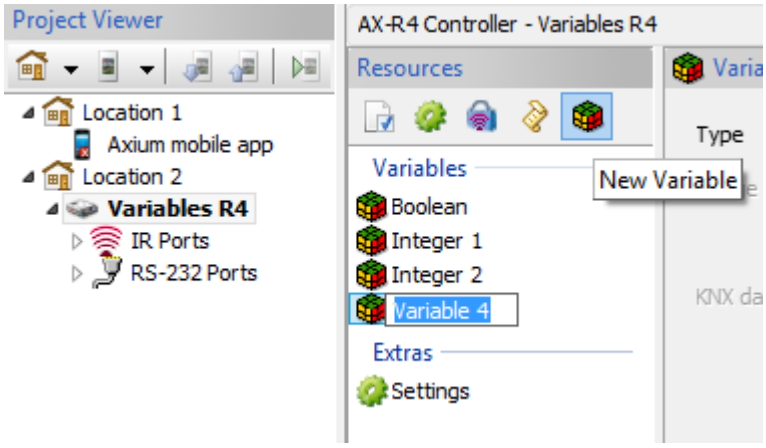


Figure 8 – New variable

A variable can be one of three types:

1. ***Boolean*** - a simple flag with the value being either true or false.
2. ***Integer*** - a whole number. The ***Range*** property defines the minimum and maximum values that the variable can be set to and only applies to ***Integer*** variable types.
3. ***String*** - a sequence of characters.

If the ***Read Only*** option is not enabled, variables may be modified by the mobile devices when implemented to do so, but if the variable is set to ***Read Only***, the variable may only be modified by the Controller itself. There are no functions in the designer for a mobile device to modify a ***String*** variable, so it is effectively always set to ***Read Only***.

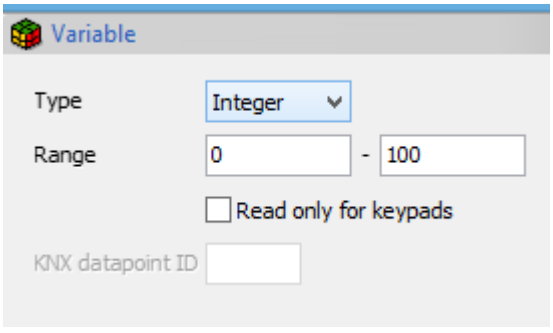


Figure 9 – Variable properties

The value of a variable can be presented on the mobile app by adding the variable to the ***Variable*** action property of a ***Button*** or ***Volume Box***. In the designer, ***Volume Box*** represents both sliders and readouts.

Setting a ***String*** variable to a slider will have no effect. When an ***Integer*** is added to a slider and the variable is not ***Read Only***, then adjusting the slider will adjust the variables value.

A button however, must set a button class from under the **Axium -> Variables** submenu in order to modify the ***Integer*** value.

A ***Boolean*** variable is toggled as the action unless the variable is set as ***Read Only***.

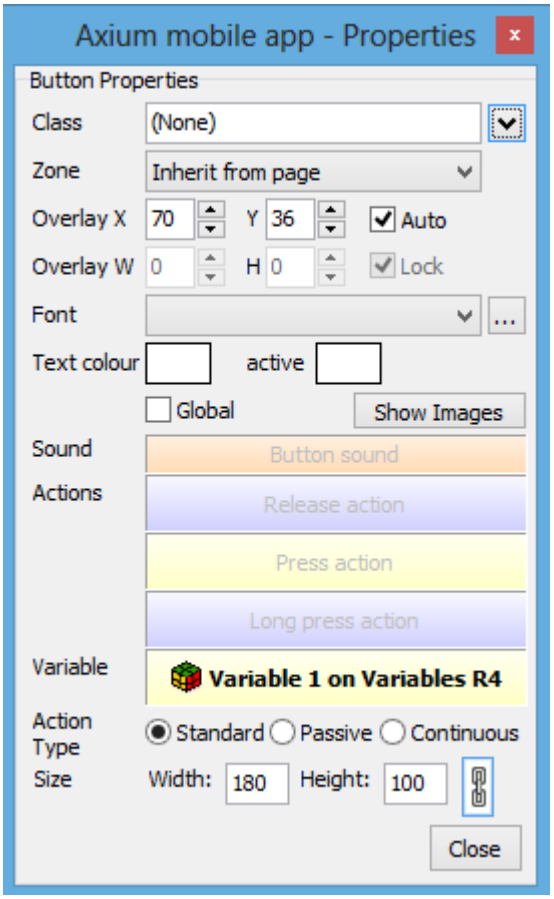


Figure 10 – Variable on button properties

Applet Design

Applets are mini applications which provide and enhance interaction with specific types of equipment or services. The following is a list of currently implemented applets:

- Axium Media Player
- Integra Receiver
- ReQuest
- Sonos
- Bticino Lighting
- CBus Lighting
- Vantage Lighting
- ELK M1 Alarm
- Schedule Event Setup
- Variable Display
- WebView

Each has its own unique properties and will be documented in its own section.

Applets are created by right clicking the layout designer, and selecting one of the listed applet types from the ***Add new applet*** submenu. A frame for the applet is then placed onto the layout, which contains a rudimentary display of how the applet may appear when it is used. Here are some common properties across all applets:

- ***Font*** – creates or selects the font used by the applet for displaying text.
- ***Text colour*** – selects the colour of most text displayed by the applet.
- ***Background colour*** – selects the colour of the background when the transparent option is not selected.
- ***Transparent*** – selects between transparent and opaque backgrounds.
- ***Device*** – selects the device to be controlled by the applet. Depending on the applet, the device may be either IP based or RS232 based. When an IP device is used the mobile device will directly communicate with the device in question, whereas if an RS232 device is used, an Axium Controller will delegate on behalf of the mobile device.

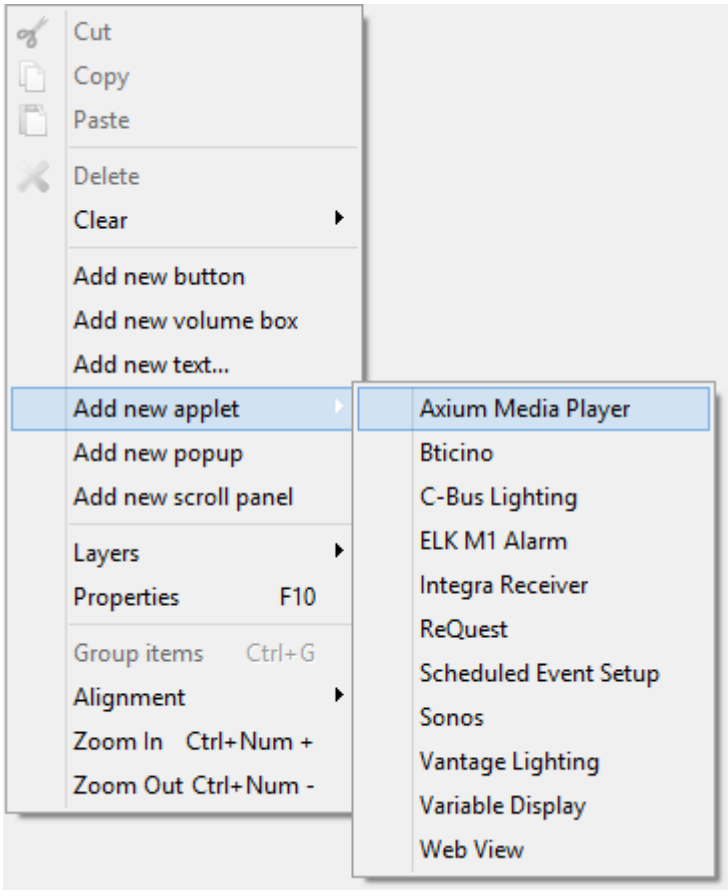


Figure 11 – Add new applet

The concept of applet instances is an essential part of the way applets are managed and function. Two applets frames with the same instance share the network connection and the execution state of the applet. Two separate frames on the other hand will not share anything between each other. Whenever applets are to be spread across multiple pages or orientations, they should be the same instance to ensure that the state and connection of the applet is retained upon a page transition. If the same applet instance is not present on a new page, then the applet will be closed and all resources freed. This will impact the user experience due to overheads and connection delays if not designed for properly.

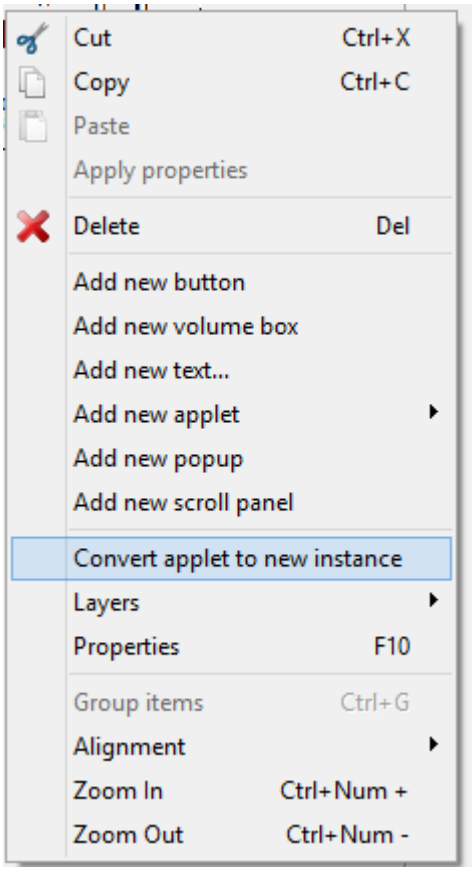


Figure 12 – Changing the instance of an applet

Some applets such as Integra Receiver and Sonos, can use multiple instances of an applet on the same page. These are unique and this should not be done for other applets or the behaviour governing which is used and displayed is undefined. The Sonos and Integra applets use the applet instance on another level, associating it with different modes making this possible. Look at specifics in their respective sections.

Applets can be morphed into existing instances of the same applet type by selecting an entry in the **Applet** drop down box in the applet properties. There you will also have the option of converting the applet into a new instance, copying the current properties as a template for the new instance.

Applets are identified by what applet instance they are; by checking the name and core properties of the applet.

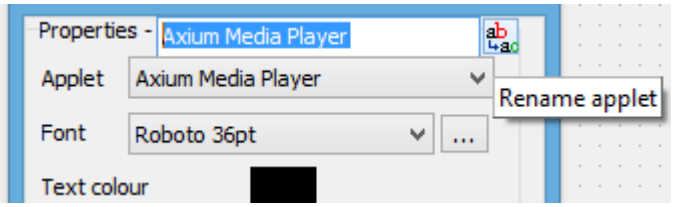


Figure 13 – Renaming applet

Most applets rely on or at least complemented by buttons, sliders and readouts. Buttons have a ***Class*** property which when set to an entry from the ***Applets*** submenu, causes the button to belong to the applet instance of that type on the layout. The instance is automatically chosen based on the type of applet, but in rare cases there may be multiple instances of the same applet on a page. In this case the button must be positioned within the frame of the applet in order for the button action to be recognised as for that specific applet. Sliders and readouts use this technique as their only means of determining which applet they are assigned to.

The class of a button defines what action that button will have with regards to the applet. Any action associated with the class is performed on the release event of a button.

There are many examples of the various applets and their associated buttons and sliders in the ***Applet Examples Hires*** gallery.

Performance considerations:

In ordinary circumstances the applet will close when a page transition occurs and the applet is determined to not exist on the new page. The result of this is the applet needing to reconnect again when going back to the page with the applet, in addition to any state that the applet was in being lost. A common design approach when using applets is to make the applet persistent across some or all pages of a design. The purpose of this is to retain the applet state and/or prevent the costly loading times where the applet initially connects to the device it is controlling.

In setup applet persistence across pages without requiring the applet display content on a page, the applet frame should be sized down to be as small as possible. When the applet frame is sized 20px or less in any dimension it will be set to a mode which will prevent rendering which saves resources as well as stopping undesired artefacts on the screen. When available, the applet frame should also be set to a view mode which will not ordinarily perform drawing. Examples of this are the "**Lite Version**" option for the Integra Receiver applet, the "**Volume Frame**" view type for the Sonos applet, and the "**No Display**" view type for the Scheduled Event Setup applet. The reason for this is largely a matter of principle but also because the internal logic of the applet might take into account which applet frame types are available during its processing. By choosing the view type which does not display anything, the applet execution may be done more optimally.

Axium Media Player

The Axium Media Player applet is a straight forward media player implementation. It does not use the device property like many other applets do; instead it directly uses the amplifier used by the design in the project. In order for the applet to function correctly, the zone of the applet must have its current source set to ***Media***. For this reason the Axium Media Player applet is typically situated on a page accessed by a button which sets the source to ***Media*** before loading the page.

The media shares to be listed by the applet are defined in ***Amplifier Settings -> Media Servers***. The username and password must match an account which has access to the share. For a share that everyone has access to, the default username guest and a blank password are normally used.

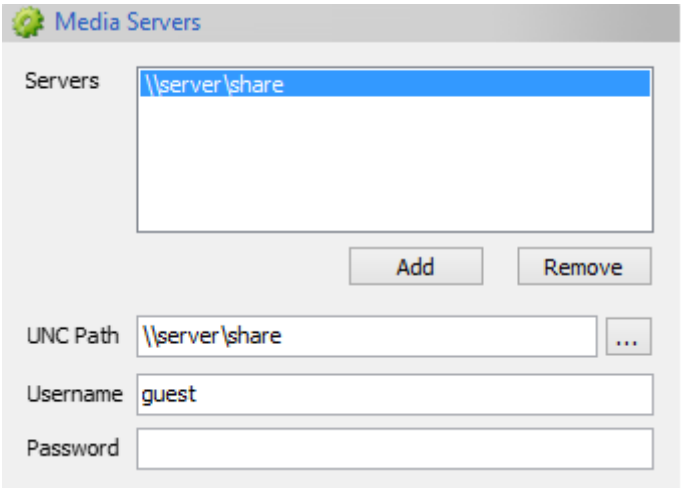


Figure 14 – Axium Amplifier Media Servers

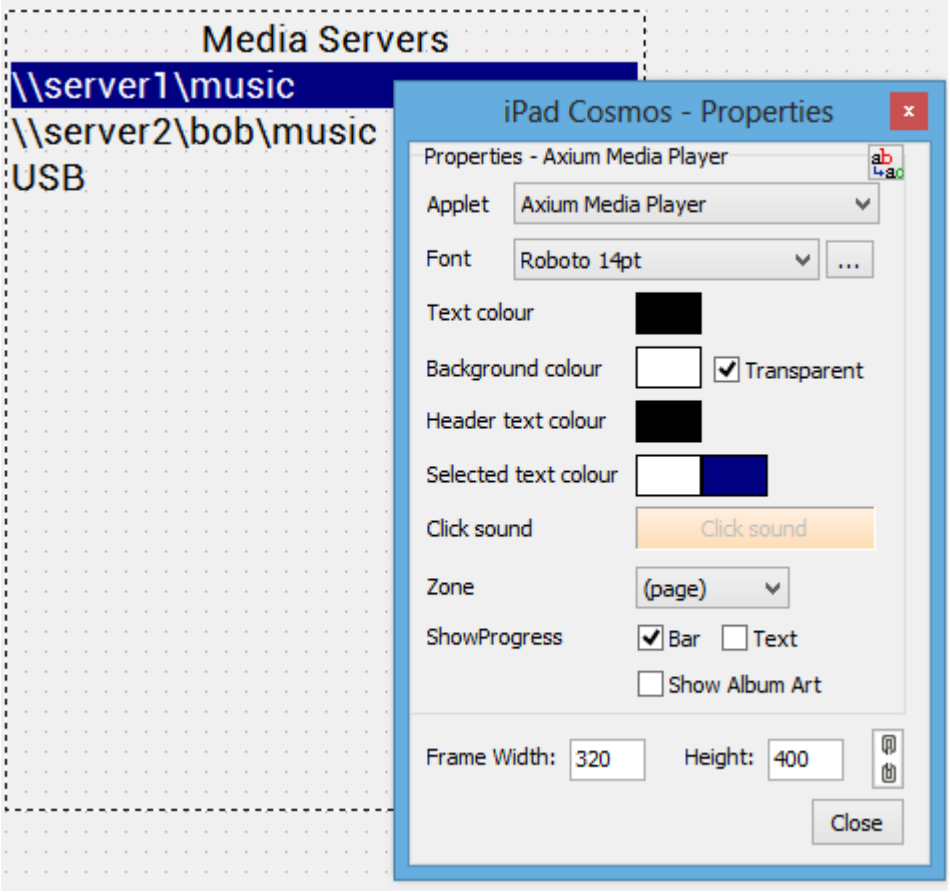


Figure 15 – Axium Media Player

Integra Receiver

The Integra Receiver applet allows controlling as well as receiving metadata and feedback from Integra AV Receivers. In many cases only the control functionality is required; in such cases the ***Lite Version*** box should be checked. When in this mode, the applet will function as normal but the applet will not display anything. It is possible with the Integra applet to mix 2 applet frames of the same applet instance on a single page when one of the frames has the ***Lite*** property set. This is very useful in the case where metadata is desired but a volume control or readout is required that is unable to be positioned in a location within the boundary of the standard applet's frame.

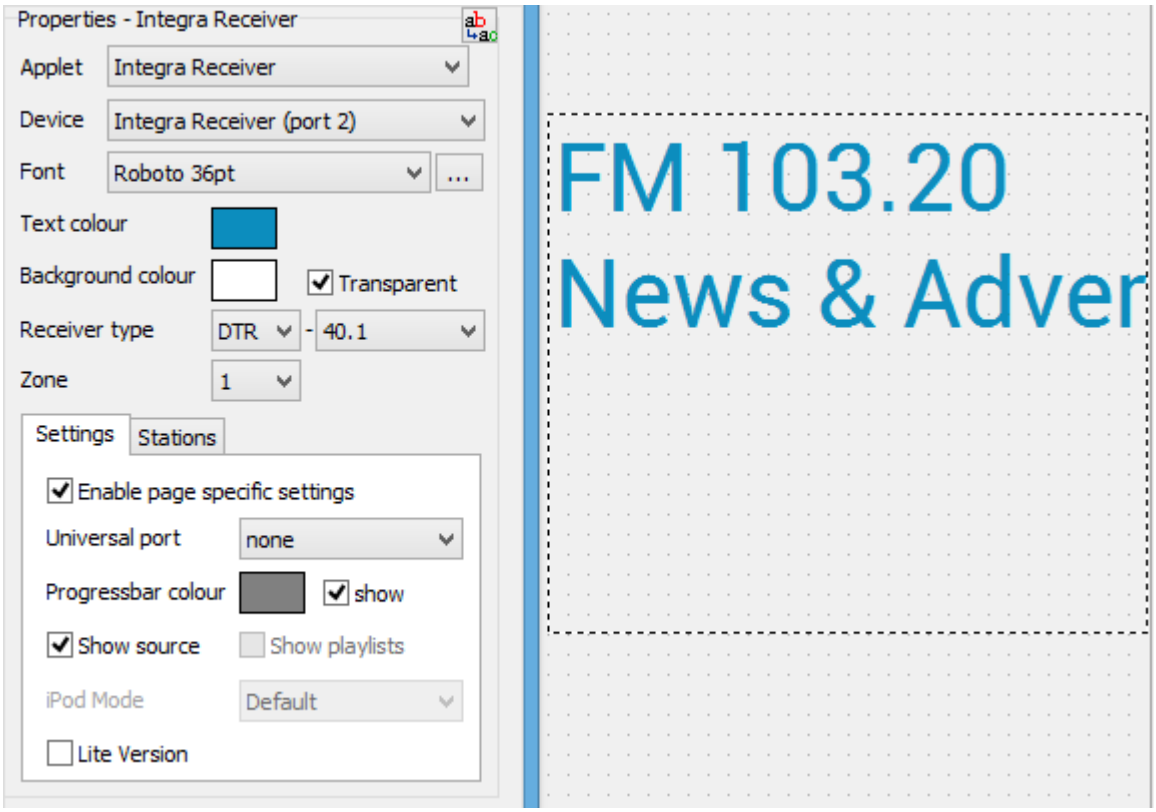


Figure 16 – Integra applet

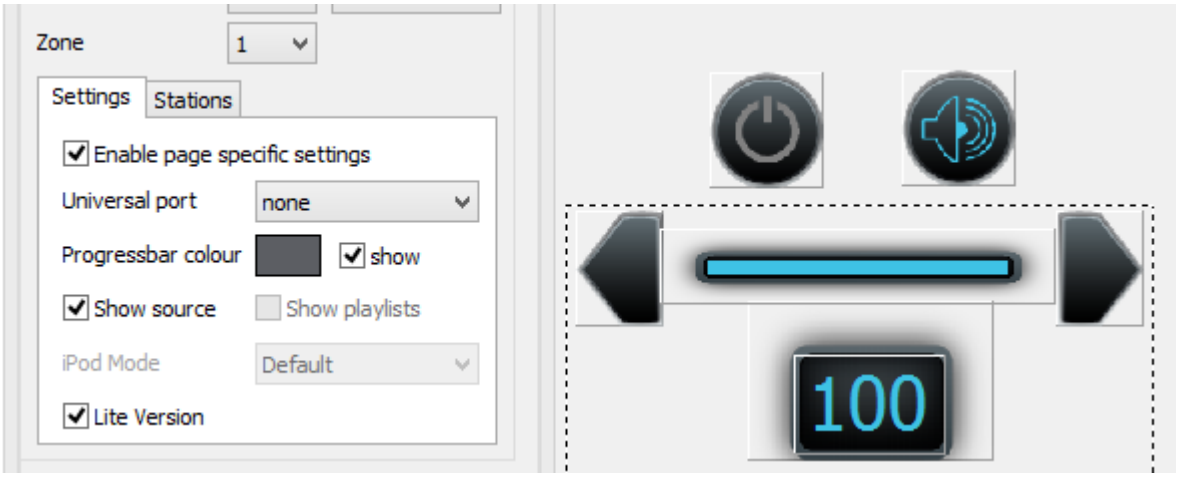


Figure 17 – Integra Lite applet

Sonos

The Sonos applet is a more advanced applet that requires some setting up in order to achieve a working solution. Firstly unlike other applets, the Device property does not need a port number associated with it. The IP address / URL is the only required field for setting up this device. The Sonos device itself should be configured and set up using the standard Sonos software prior to attempting to use this applet for control.

Another point of difference with the Sonos applet is that its various functionalities are presented by using four different view types:

1. ***Now Playing***
2. ***Queue***
3. ***Content Browser***
4. ***Volume Frame***

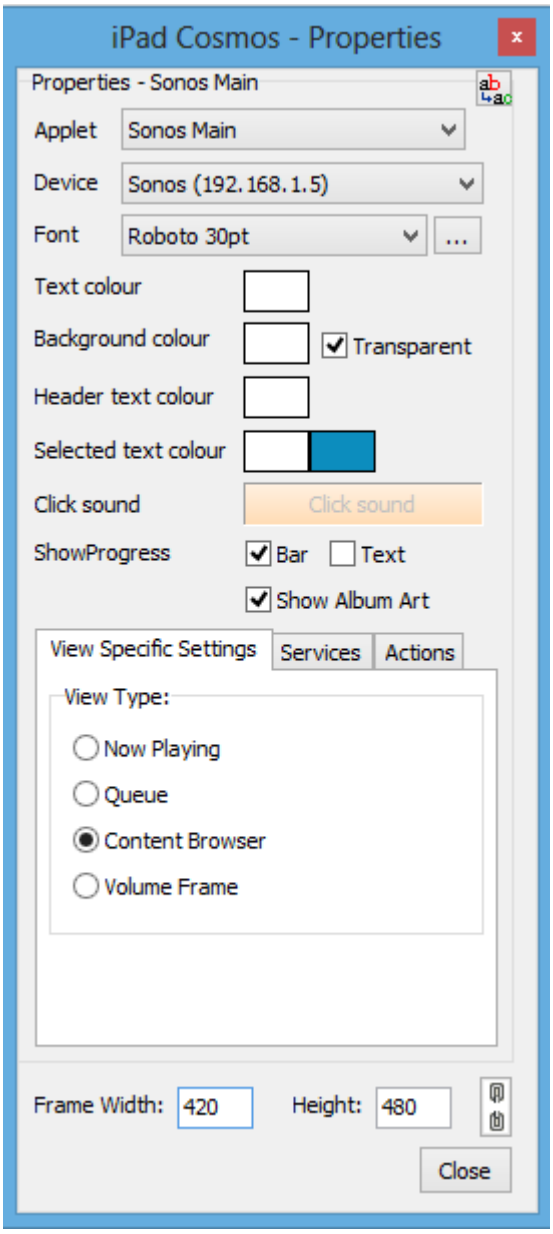


Figure 18 – Sonos properties

In order to achieve complete functionality, the first three of these types will need to be used in the design, and all need to be the same applet instance. Refer to Applet Design for more on this. The volume frame offers more fine-grained control over the positioning of any volume control or readout.

The different view types may be mixed and matched together or separately, across a single page or several pages; there is no limitation on this. One important thing to note is that some functionalities of a view type require the other view types to be activated, which may not initially be the case when the applet frames are spread across multiple pages. For this reason adding the three view types together on the same page encapsulated by a scroll panel or a few popups is recommended. This way all applet frames are initialised together and no functionality will be missing.

Currently the Sonos applet supports three external services: **Spotify**, **Pandora**, and **MOG**. These services need to be configured using the Sonos setup software prior to using them with the applet. Each service can be added by enabling it and entering the authentication details for the account in the ***Services*** tab of the Sonos properties. These details are encrypted and are secure. They are required in order for the mobile device itself to connect to the service; the Sonos device will have its own setup details for its interactions with these services.

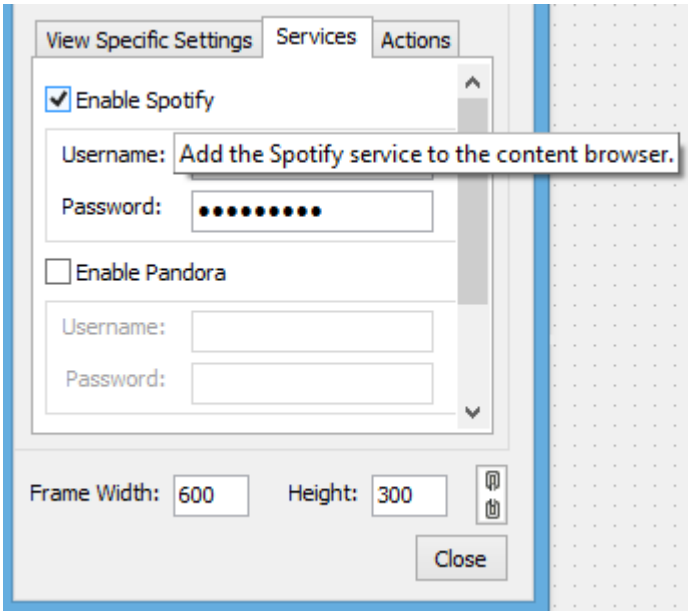


Figure 19 – Services tab in Sonos properties

The button classes for the Sonos applet are organised within four groups, which pertain directly to the view types.

Navigator Up through to ***Back***, are used to control the Content Browser view type.

Playlist Up through to ***Select***, are used to control the Queue view type.

Play through to ***CrossFade***, are used to control the Now Playing view type.

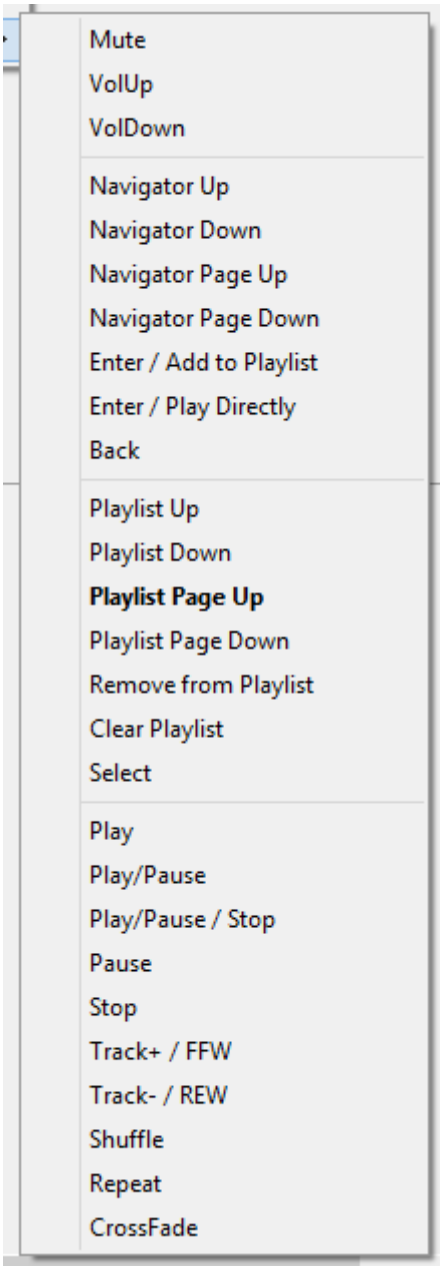


Figure 20 – Sonos button classes

ReQuest

The ReQuest applet is a simple media player applet similar in functionality to the Axium Media Player applet. Due to constraints imposed by the ReQuest control API, the playlist is not scrollable.

Vantage Lighting

The Vantage lighting applet brings simple integration of lighting control. The configuration is very straightforward which mostly involves setting the appropriate VID and any options for tasks.

Bticino Lighting

The Bticino lighting applet allows for simple lighting control. Some set up is required on the Bticino side to enable control via this applet. Refer to the Bticino documentation regarding ***"IP address enabling mode"*** for more information.

CBus Lighting

The CBus lighting applet offers more complex lighting control. The applet setup is fairly straightforward with some configuration options available to cater for more advanced purposes.

When using the CBus lighting applet it is recommended to use an RS232 - CBus PC Interface connected to an Axium Controller as the device for the applet. This is opposed to the option of using the Ethernet interface for CBus. The reason for this is that the CBus Ethernet interface only allows one concurrent connection and does not correctly detect when a used client disconnects. This makes for the very real possibility of complete loss of control in these circumstances. The issue is compounded by the fact that the mobile devices use Wi-Fi to establish Ethernet connectivity and any drop in connection could trigger this serious problem to occur.

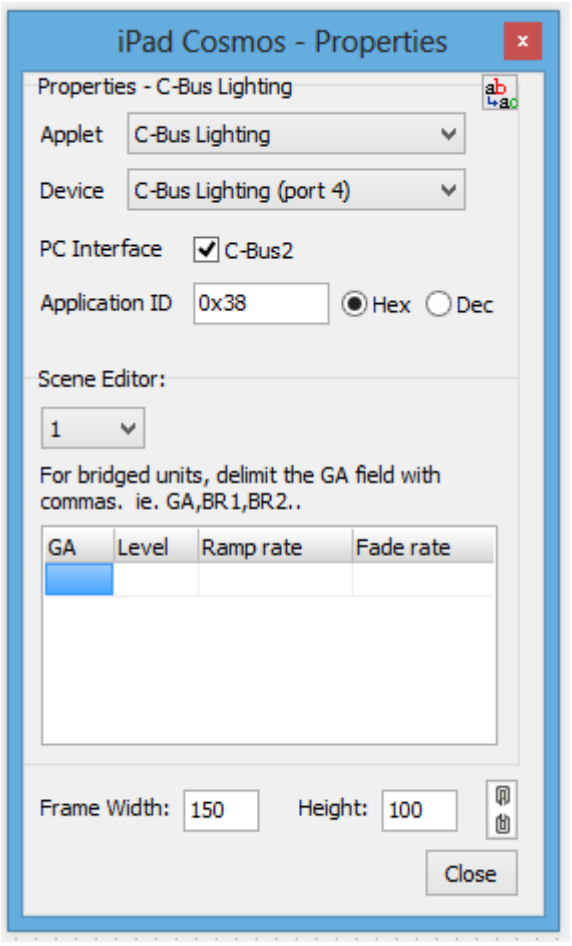


Figure 21 – CBus properties

ELK M1 Alarm

The ELK M1 Alarm applet is a straightforward applet to manage ELK and Ness M1 / EZ8 alarm systems.

The **Area** property defines the area which the applet will control. Only zones that are associated with this area can be controlled by the applet. If more areas are needed to be controlled then an additional applet instance will be required for such handling.

The **Baud rate** property is used when using rs232 serial as the means of connecting an Axium controller to the alarm system. The value should match whatever the alarm system is set up for. If IP control is used then the value of this property is irrelevant.

The **Pin Length** property must match the size of the configured pin code for the alarm. When the number of required digits is entered with the button classes **0** to **9**, the arming operation or any other privileged operation that is being performed will be sent to the alarm system to handle.

The **Temp. unit** property is used in conjunction with any buttons using the **Zone temperature** button class which outputs the sensory information from a temperature sensor probe configured as a specific zone.

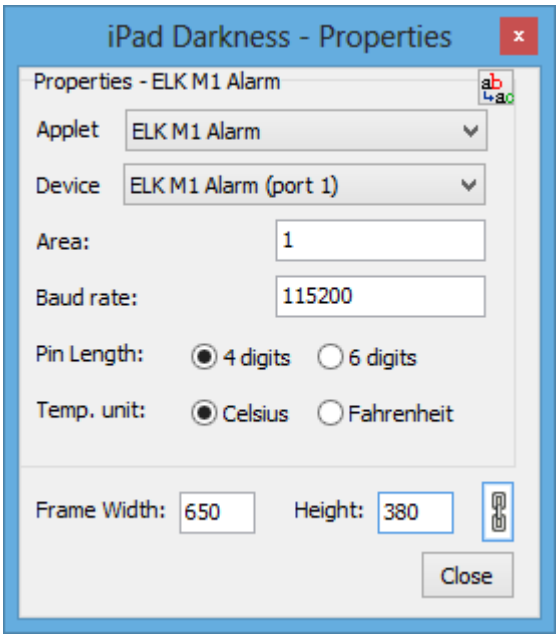


Figure 22 – ELK alarm properties

Variable Display

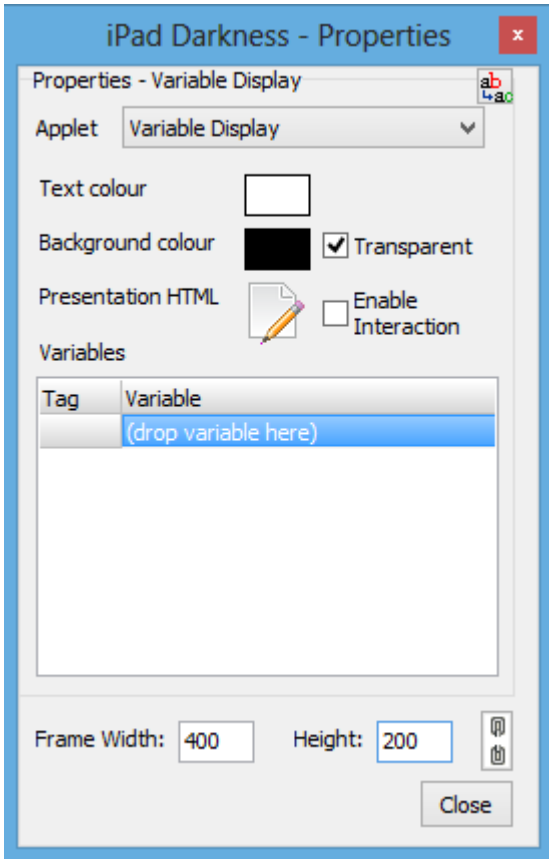


Figure 23 – Variable Display properties

The Variable Display applet is a rather unique applet type offering flexible presentation and design possibilities. The purpose of this applet is to display information related to the values of variables. This is achieved by using HTML to define a layout and using the javascript library sprint.js to provide simple manipulation of data into various formats.

Variables are added to the applet by dragging and dropping a variable from an Axium Controller onto the variable display properties grid. The tag of the variable is given which is the array name and index used in the user defined script in the HTML layout.

Since this applet is HTML based the applications of this applet are vast and extend beyond the purposes if solely displaying variables. This applet could be used for example to display animated GIF images, which are otherwise unsupported. The **Enable Interaction** option will allow for the user to interact with the html as an ordinary webpage allows.

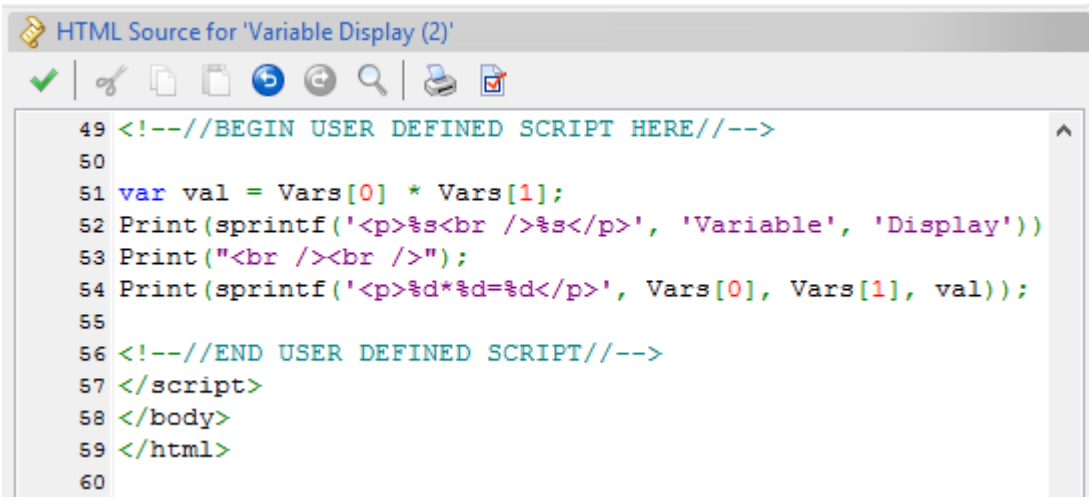


Figure 24 – Editing the source HTML for the Variable Display

WebView

The webview applet enables the Axium mobile app to access URL resources. The content could range from a web page to images and videos. A common application for this applet is to display live streams or updates from IP cameras and web cameras. The ***Mode*** property of the webview has four options which are tailored toward providing the most optimal experience and efficiency depending upon the type of content. The modes are:

1. ***Default WebView*** - a standard web browser control which offers a full web experience. Useful for navigating websites and streaming videos.
2. ***Static Page*** - a standard web browser control but with no user interaction. Great for displaying a page and restricting navigation and input.
3. ***URL Image*** - a lightweight image display. The most efficient way to display a static image.
4. ***MJPEG Stream*** - similar to URL image, but used for a stream of motion JPEG images which is a common format for IP cameras.

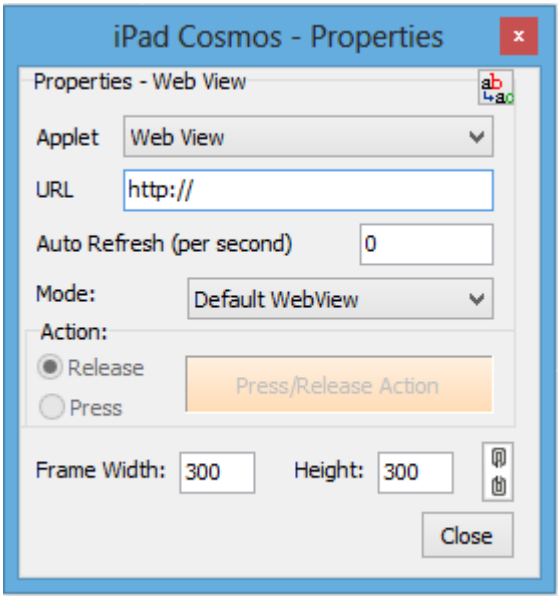


Figure 25 – WebView properties

The ***Auto Refresh*** property defines the frequency for which the webview will perform a refresh of the content. When ***MJPEG Stream*** is used, this is the frame rate for the stream.

The ***Action*** property allows a non-default webview to register a press or release functionality. This can be useful for example to display thumbnails of a web camera image, and have the action display a larger version of the image.

Scheduled Event Setup

The scheduled event setup applet creates the provision for manipulating scheduled triggers. These triggers are created on an Axium Controller and have much functional flexibility with features including repeats, secondary actions, sunrise/sunset offsets, and customisability for the days of the week on which the trigger will occur.

Each time or offset input field is designed by implementing multiple applet containers each representing the same applet instance. The **View Type** property for each container is set to define which input field to display. The **No Display** view type is a special type with no functionality. It is useful for defining which page items are affiliated with a particular applet instance. This is essential when designing a layout with multiple applets instances on a page.

When the view type is set to either '**From/At Offset**', '**Until/Off Offset**' or '**Repeat Interval**', then the **Unit** property will become available to set. This property defines what measurement of time to use when displaying the value. The same property should be set on the affiliated button class used to increment or decrement the value.

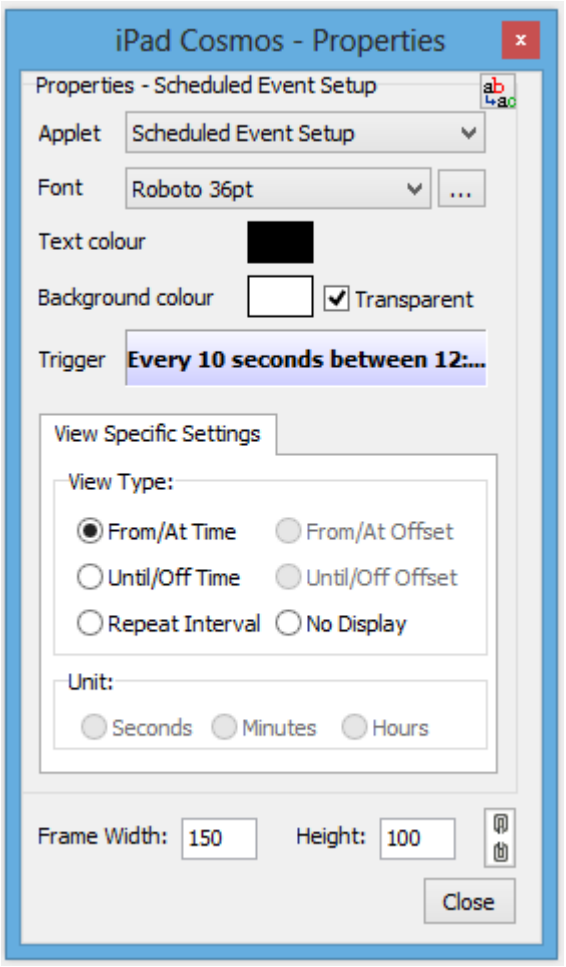


Figure 26 – Scheduled Event Setup properties

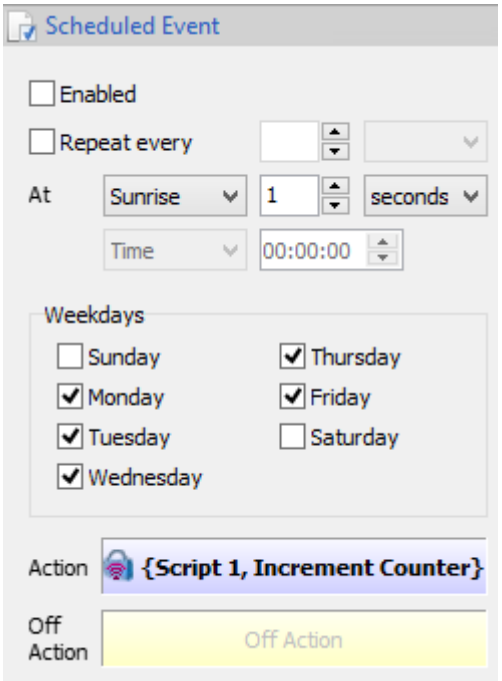


Figure 27 – Scheduled Event creation

Upload Procedure

The process of uploading a design to a mobile device involves two phases.

1. ***Discovery*** - When the upload toolbar button is clicked, a popup windows is displayed which will list devices running the Axium app that are found on the network. The mobile device must be in ***Update Mode*** which will make it respond to the discovery broadcasts of Axium Design Portal. The mobile device will respond every time the app is entered or refocused while the upload popup window is open.
2. ***Upload*** **-** When a device is selected and the ***Upload*** button on the popup window is clicked, the actual upload will commence.

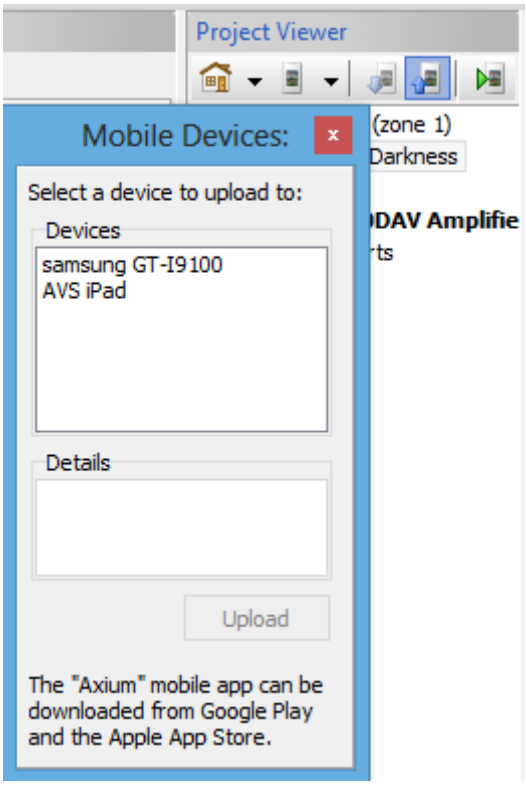


Figure 28 – Upload box

FAQ / Troubleshooting / Common Issues

1. My device does not appear in the upload box in Axium Design Portal.

There a number of possible reasons for this:

- The discovery mechanism uses broadcast over the local network, so ensure that the mobile device and PC are both accessible within the same subnet. Routers do not pass broadcast traffic over WAN ports.
- If the PC is connected to multiple networks, ensure that each is on a unique subnet.
- Ensure that ***Update Mode*** is enabled in the settings on the mobile app.
- Ensure that only one instance of Axium Design Portal has the upload box open at a time.

If all else fails try terminating the mobile app, toggling Wi-Fi, or rebooting the device.

2. Sending commands to an Axium Controller or Amplifier doesn't work and a message displaying "Unknown Object" appears.

This error message is a response from the Axium Controller or Amplifier signifying that the command it received does not exist.
The most common reason for this is that the Axium Controller or Amplifier has not been uploaded to after the command was referenced. Whenever changes are made requiring another upload, the device in the project viewer will become highlighted.
Another possibility for this message is the presence of a duplicated device on the network creating conflict. This can accidentally be done by uploading the same device configuration to more than one Axium Controller or Amplifier.

It is important to note that when commands are added to an Axium Controller or Amplifier, they are only uploaded to the device when something in the project uses them. There is a common misconception that because the commands were added and an upload was performed, that the device is set up. Whenever a command is referenced for the first time, it will require another upload in order for that command to actually be present on the Axium device.

3. Occasionally when using my iOS device, commands sent to an Axium Controller or Amplifier no longer work.

There is a bug prior to iOS 7 which causes the multicast DNS resolution to stop working. The Axium protocol uses this technology in order to identify devices on the network. Whenever this bug is triggered, the mobile device is unable to determine the IP address of the Axium Controller or Amplifier resulting in no communication between the devices.

When this problem occurs, toggling the Airplane Mode switch in Settings restores the Apple DNS service to a functional state.

A permanent workaround is to set static IP addresses for the Axium Controllers and Amplifiers so that the DNS service is not required.

4. My iOS device does not connect or takes a very long time to connect to an Axium Controller or Amplifier.

Unfortunately some wireless access points -especially very old ones, do not handle multicast traffic well. In the event that one of these bad access points is in use, standard Apple discovery mechanisms will fail also. The best solution is to upgrade to a decent access point, but setting static IP addresses for all Axium hardware will also workaround this issue.