PIXI
LAB

Blocks®

VERSION 7.2

CONTENTS

1. Introducing PIXILAB Blocks 3

2. Getting Started 5
   - Mac 5
   - Windows PC 6
   - Accessing the Editor 7
   - Linux 9

3. Concepts 11
   - Spots 11
   - Blocks 11
   - Managed Subsystems 12
   - Tasks 13

4. Spots 14
   - Spot Settings 15
   - Display 15
   - Visitor 21
   - Location 23
   - WATCHOUT Spot 25
   - Spot Group 25

5. Blocks 27
   - Block Protection and Templates 30
   - Slideshow 31
   - Composition 34
   - Book 37
   - Child Replication 38
   - Stack 39
   - Scroller 40
   - Grid 41
   - Attractor 41
   - Tag Selector 43
   - Media File 44
   - Media URL 46
   - Camera 47
   - Panorama 48
   - 3D 49
   - Magazine 51
   - Text 52
   - Locator 55
   - Synchronizer 58
   - Web 60
   - Widget 61
   - Reference 62
   - Live Video 62
   - Subtitle 64
   - Schedule 64

6. Timeline 66
   - Tracks and Cues 67
   - Timeline Commands 72

7. Behaviors 74
   - Appears 74
   - Disappears 74
   - Move 74
   - Mute 75
   - Play 75
   - Playing 75
   - Press 76
   - Reveal 76
   - Navigate 76
   - Rotate 76
   - Scale 77
   - Zoom 77
   - Speed 77
   - Volume 77
   - Classify 77
   - Opacity 78
   - Blur 78
   - Contrast and Brightness 78
   - Grayscale and Sepia 78
   - Hue 78
   - Saturate 78
   - Perspective 79

8. Controls and Panels 80
   - Button 81
   - Indicator 85
   - Slider 85
   - Bar 85
   - Joystick 85
   - Text 86
   - Text Input 86

CONTENTS

QR Code 86
QR Scanner 87

9. Spot Parameters 88
Interactivity with Behaviors 90
Custom Spot Properties 90
Use in Property Paths 91

10. PIXILAB Player 92

11. Visitor Data Collection 93

12. Manage 94
Users 94
Licenses 94
Mirroring Service 95
Status and Log 96
Modbus I/O Modules 97
Network Devices 99
Art-Net/DMX-512 Lighting 102
WATCHOUT 105

13. Task 106
Realm 106
Group 107
Task 108
Statements 111

14. Properties and Functions 119
Display Spot 119
Visitor Spot 121
Location 121
Spot Group 122
Timeline 122
Task 122
Modbus Channel 123
Art-Net/DMX 123
WATCHOUT 124
Schedule 126
Network 126

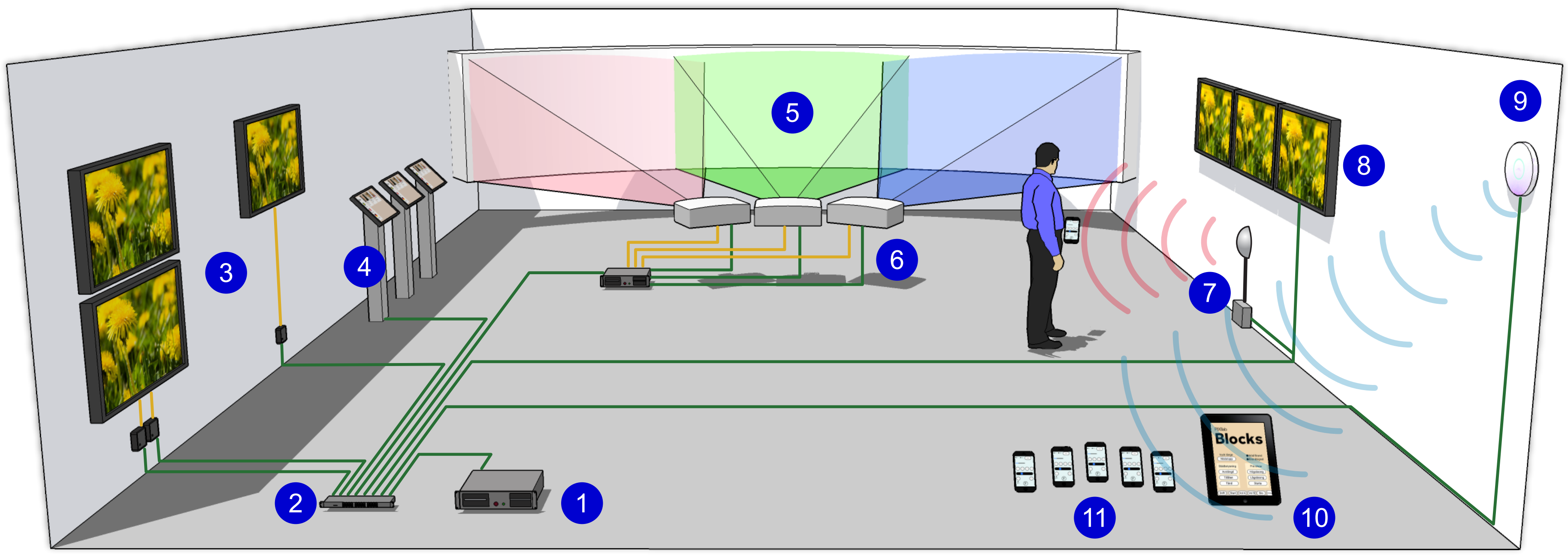
15. Authorization and Roles 127
Configurable Access Restrictions 128

A. Server File Structure 131

B. Scripting and Device Drivers 132

1. INTRODUCING PIXILAB BLOCKS

PIXILAB Blocks® is software for producing and managing rich experiences for visitor centers, museums, exhibitions and corporate presentations. It features a unique mix of content and display management, control system capabilities, interactive presentations and mobile guide functions.



Basic Blocks system and wiring diagram. Network (green), video (yellow), wifi (blue) IR/beacons (red).

The illustration above shows a typical Blocks system, containing the following notable elements (numbers referring to blue circles).

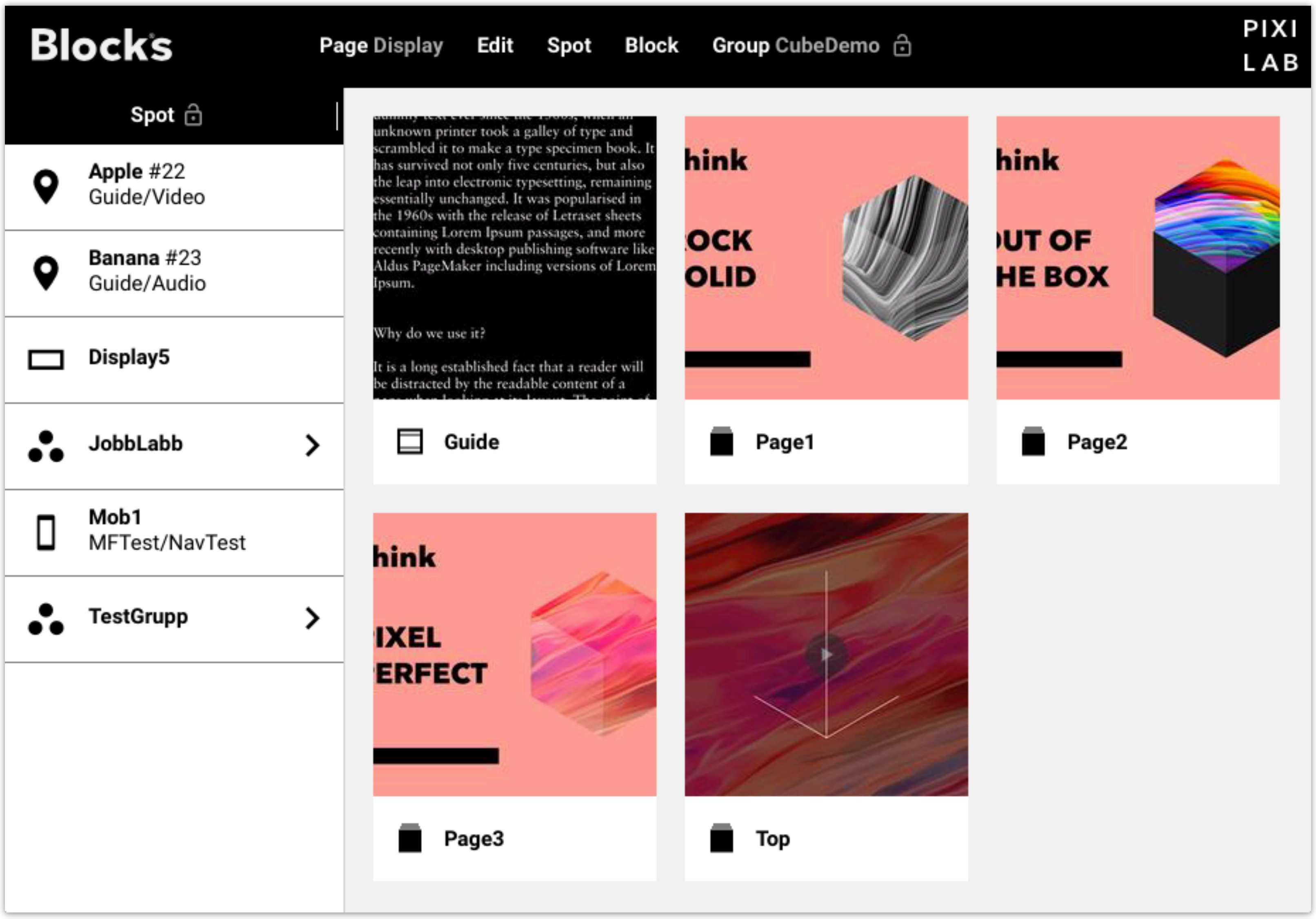
1. PIXILAB Blocks Server.
2. Wired network.
3. Displays with separate players.
4. Interactive touch kiosks.
5. Large format display subsystem, such as Dataton WATCHOUT®, on a curved screen.
6. Projectors and other devices controlled and monitored through the network.
7. Relays, switches and sensors (here an IR motion sensor) connect through standard interface boxes.
8. Smart displays with built-in players connect directly to the network.
9. Wifi network for internal management and/or public access.
10. Management panels for local staff use.
11. Visitors' phones or loaner devices for mobile guide applications.

Server

At the heart of Blocks is the server software. This program can run on your personal computer or on powerful, dedicated server hardware. Using your own Mac or Windows PC is convenient for production work and quick tests, as well as for running smaller projects. Dedicated server hardware, running under Linux, is available for larger and more demanding installations.

Editor

Your presentations are built and edited using a regular web browser that connects to the Blocks server over the network. Here, content – such as images, video and sound – is combined into reusable and composable blocks. Content is then assigned to displays or mobile devices, collectively referred to as *spots*.



unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

Why do we use it?

It is a long established fact that a reader will be distracted by the readable content of a

The Display page of the editor shows displays and other spots as well as the content blocks.

Spots

Locations of interest, such as displays, theaters, geographic locations or positions inside a building, are commonly referred to as *spots*. The spots are listed on the left hand side on the Display page, (see above) and can be arranged into groups and sub-groups as desired to simplify navigation and programming.

Spots also include mobile devices, which may be a permanent part of the system (such as an iPad used as an operator panel), or temporary (such as visitors' mobile phones).

Blocks

Just like LEGO bricks, blocks can be combined in an almost infinite number of ways. This lets you build anything from a simple digital-signage-style slide show up to an interactive kiosk or a mobile guide system with video and audio playing in sync across multiple devices.

For example, use a Composition block to arrange images and buttons to create an attractive user interface. If your design is complex, use a Book block to spread multiple Composition blocks across several pages. To tell a story, use a Slideshow block with a couple of images or short video clips. For an interactive visitor kiosk, use an Attractor block to show an attract loop until a visitor approaches the kiosk.

2. GETTING STARTED

This chapter describes how to get Blocks up and running for producing and testing purposes, and for running the finished installation. While you can use your Mac or Windows laptop for production and basic tests of all Blocks functions, we recommend a Linux-based, dedicated server for running the final installation.

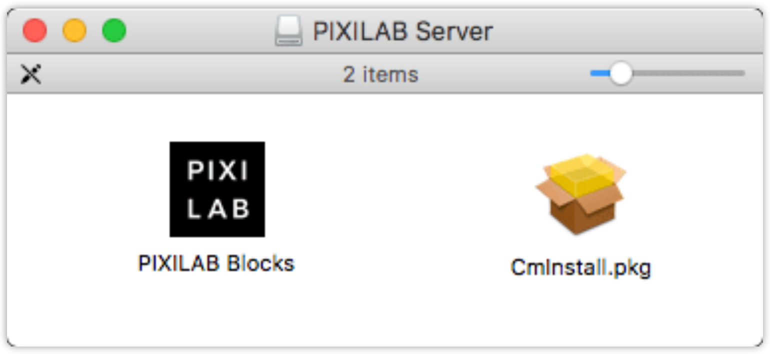
Mac

Using a laptop or desktop computer makes it easy to work with other content production software for images, video and audio. While you can access a server from any computer over the local network, having Blocks installed on your laptop means you can work locally with content production as well as testing spots.

Installation

To install Blocks on your Mac, do as follows:

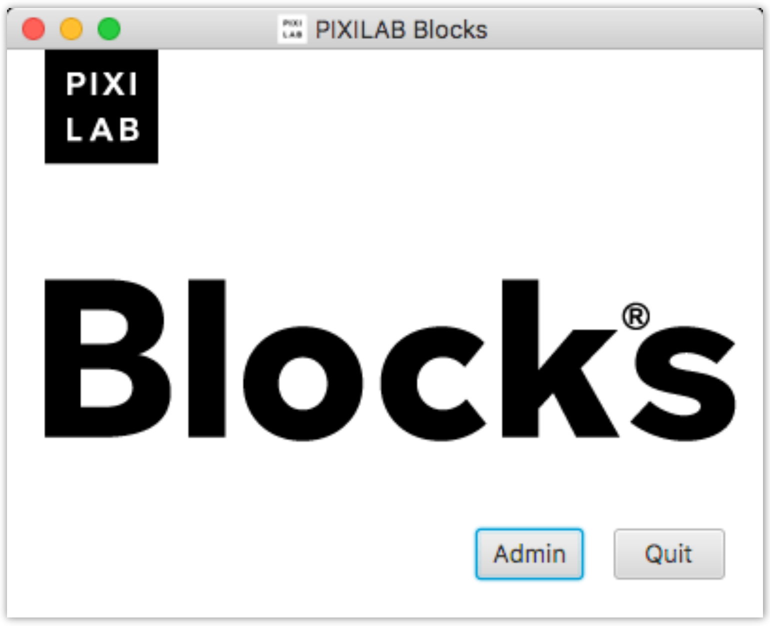
1. Download the PIXILAB_Blocks.dmg installer from https://pixilab.se/docs/blocks/macos.
2. Open the DMG file, read and agree to the license agreement.
3. After opening the disk image, you'll see a window like the one shown below.
4. If this is the first time you install Blocks on this computer, run the CmInstall.pgk installer. This installs the license key driver. You generally don't need to repeat this step when updating to a newer version of Blocks.
5. Drag the PIXILAB Blocks application icon to your Applications folder, or other suitable location.
6. Connect a PIXILAB license key to a USB port on your computer, or install the license credentials file provided if using a cloud-based license (requires internet connection).



Mac installer, after opening PIXILAB_Blocks.dmg.

Launching the Blocks Server

Locate “PIXILAB Blocks” in your Applications folder, or wherever you installed it. To simplify future use, you may want to drag the application icon to the Dock. Then start Blocks. This opens the window shown below. Since all administration of the server is done over the network, there isn’t much to see in this window. Open the Blocks Editor by clicking the “Admin” button.



PIXILAB Blocks server window.

Open the Blocks Editor using the “Admin” button in the server window, then proceed as described above under “[Accessing the Editor](#)” later in this chapter.

Windows PC

Using a laptop or desktop computer makes it easy to work with other content production software for images, video and audio. While you can access a server from any computer over the local network, having Blocks installed on your laptop means you can work locally with content production as well as testing spots.

Installation

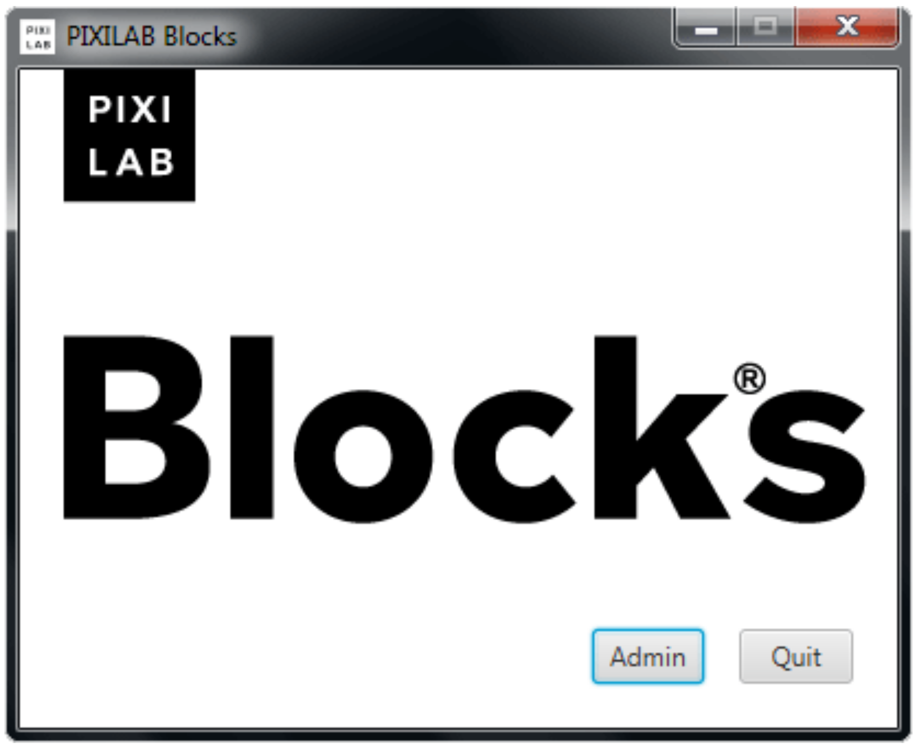
To install Blocks on your Windows PC, do as follows:

1. Download the PIXILAB_Blocks.zip installer from https://pixilab.se/docs/blocks.
2. Right-click the PIXILAB_Blocks.zip file and choose “Extract All” to unpack the ZIP archive.
3. A window will open with the two icons shown to the right.

4. If this is the first time you install Blocks on this computer, run the “CodeMeter Runtime.exe” installer. This installs the license key driver. You generally don’t need to repeat this step when updating Blocks to a newer version.
5. Double-click the “PIXILAB-Blocks-Installer.exe” and follow the instructions to install Blocks.
6. Connect a PIXILAB license key to a USB port on your computer, or install the license credentials file provided if using a cloud-based license (requires internet connection).

Launching the Blocks Server

Click “PIXILAB Blocks Server” on the Start menu. This opens the window shown below. Since all administration of the server is done over the network, there isn’t much to see in this window. You can open the Blocks Editor by clicking the “Admin” button.



PIXILAB Blocks server window.

Accessing the Editor

Open the Blocks Editor using the “Admin” button in the server window, or by going to the server’s IP address followed by */edit*.

For example, if your Blocks server has the IP address 10.2.0.10, you should be able to access the editor by typing the following into a web browser on a computer connected to the same network as your Blocks server:

```
http://10.2.0.10:8080/edit
```

This will prompt you for a user name and password. The first time you start the server, there’s just a single user, named *admin* with the password *pixi*. You should change this password as soon as possible if the server is publicly accessible (this is done on the Manage page, as described under “Users”).

Once past the login window, you should see a window similar to the one shown under “Editor” above.

- The IP port number used by the server (8080 in the example above) may vary with your server configuration.For instance, the Linux server image is configured to use the default port 80, in which case no port number needs to be specified.

Adding a Display

You need a Display to test the content you produce in Blocks. This can be just a computer running a modern web browser, such as Google Chrome or Apple Safari. Preferably, the browser should run in full screen mode, with the address bar and other controls hidden. Assuming your Mac-based server has IP address 10.0.2.10, as in the example above, point your display computer to:

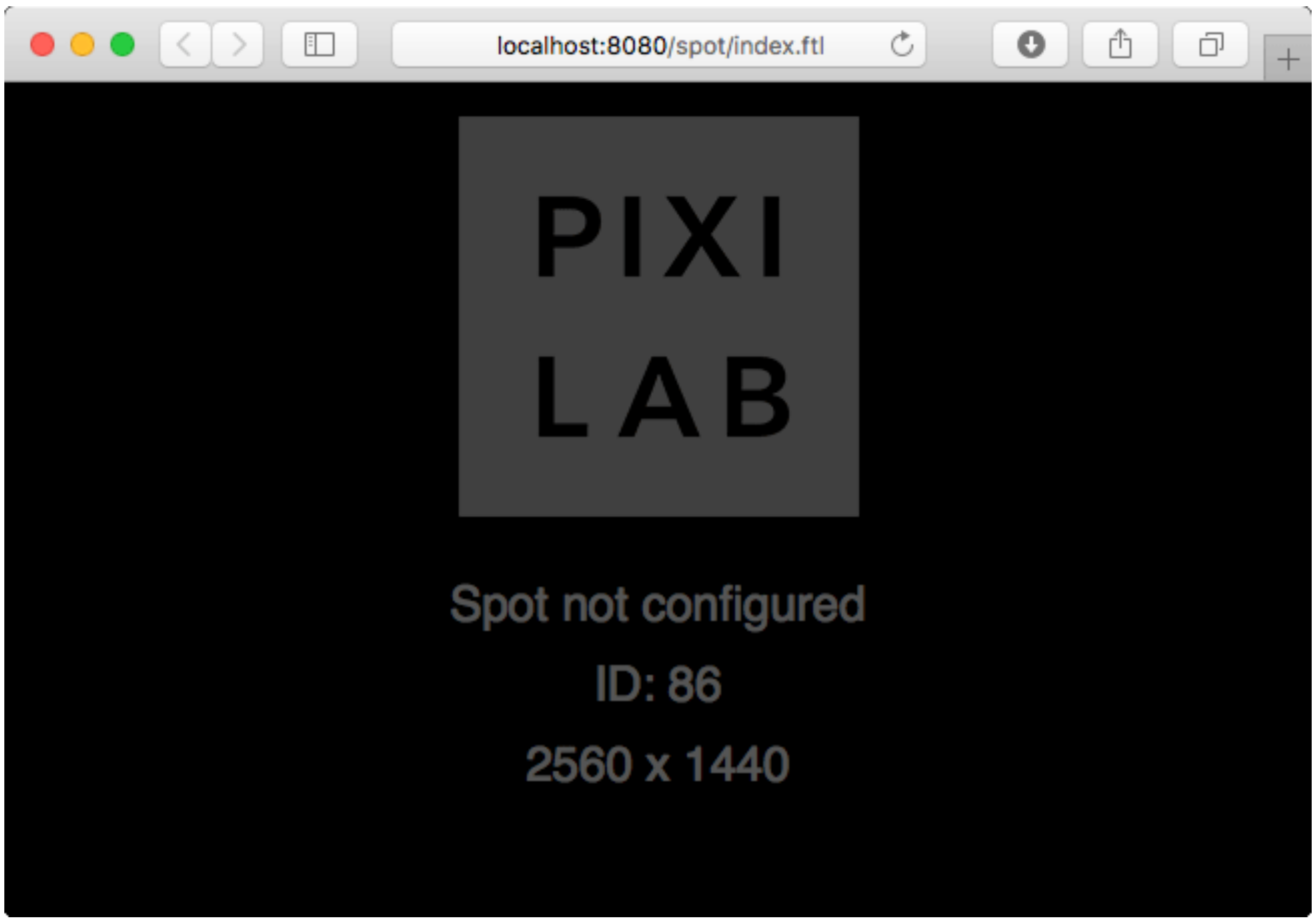
```
http://10.2.0.10:8080/spot
```

If you don’t have a separate computer available, you can simply open another browser window on your laptop, pointing it at the following address:

```
http://localhost:8080/spot
```

The name *localhost* always refers to the IP address of the local computer, so it can be used as a shorthand instead of the computer’s IP address.

You will be greeted by a predominantly black window, like the one shown below, indicating the spot's ID number and its full screen dimensions.



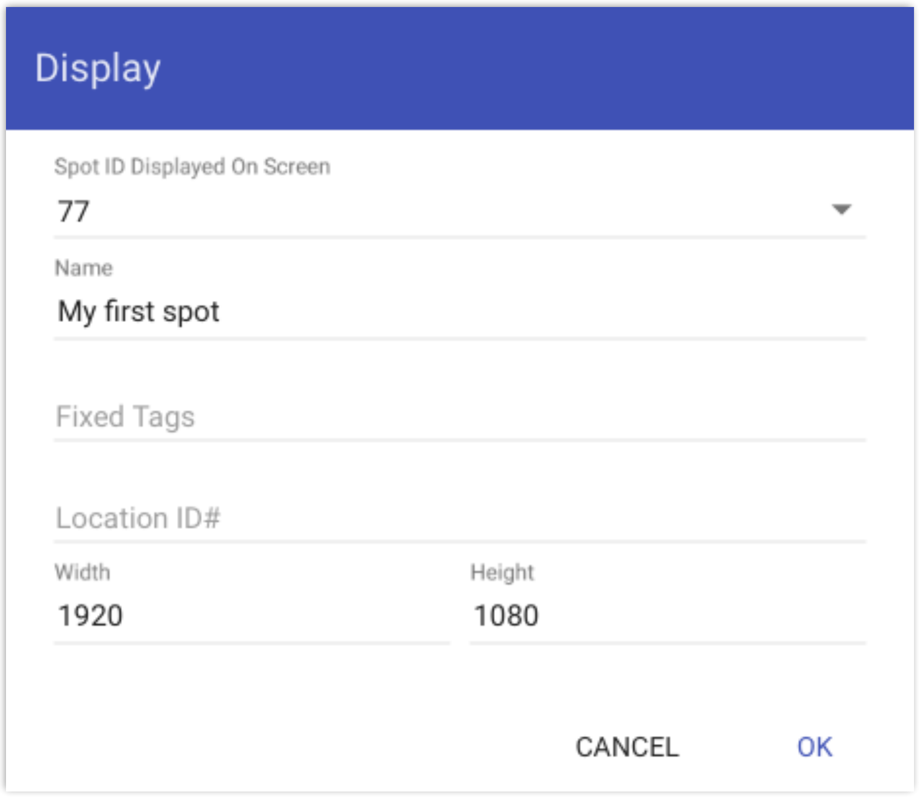
Spot not configured

ID: 86

2560 x 1440

Display before being added to the server.

Leaving the browser window on screen, return to the Blocks Editor. Choose "Add Display" on the Spot menu. Choose the ID number shown on the display as the "Spot ID", and type a name into the Name field.



Display

Spot ID Displayed On Screen: 77

Name: My first spot

Fixed Tags

Location ID#

Width: 1920

Height: 1080

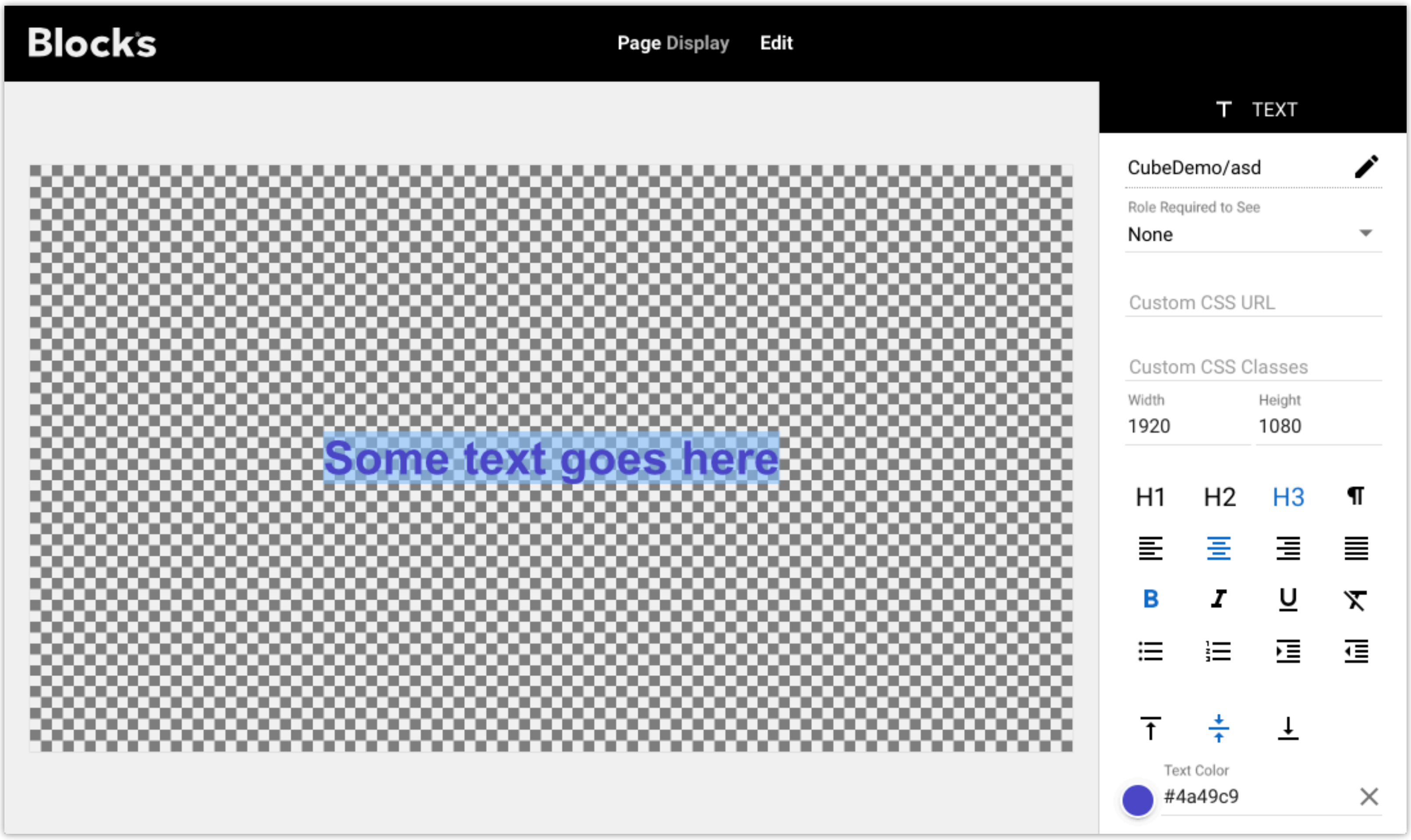
Adding a Display Spot by specifying its Spot ID.

Clicking OK in the dialog box shows the name on the Display Spot instead of "Spot not configured", indicating that the display has been successfully adopted into the system.

Displaying a Block

To make something appear on the display, you simply drag a Block onto the Spot on the Display page. If there's no block in your system, you can create a simple Text block as follows:

- Chose "Add Text" on the Block menu, giving the block a name, then click OK.
- Double-click the text to edit it, center it horizontally and vertically using the corresponding buttons to the right.
- Change the text color.
- Return to the main Display page by clicking the "Blocks" logo in the top left corner.
- Drag the newly created Text block onto the spot to make it appear on the display.



CubeDemo/asd

Role Required to See: None

Custom CSS URL

Custom CSS Classes

Width: 1920

Height: 1080

Text Color: #4a49c9

Creating a Text Block.

Linux

For fixed installations, the Linux version of the server is recommended. Unlike the Mac and Windows versions, which can be installed on your own personal computer, the Linux version of Blocks is intended for dedicated use. This guarantees the best possible long-term stability – particularly when combined with known hardware. The Linux version of Blocks is provided in the following two forms:

- Pre-installed on server hardware provided by PIXILAB or one of our authorized resellers.
- As a pre-configured *Linux image*, for installation on qualified hardware.

The first option provides you with a complete, turn-key hardware/software combination, ready to be used as an optimized system. It comes in a 19 inch, rack mountable case. The base configuration has two 1Gb network ports, but it can also be provided with a dual 10Gb card. Contact PIXILAB for further details on specifications and available options.

The Linux image option allows you to download and install a pre-configured server on your own hardware. It's tested and guaranteed to work on NUC small form factor computers, with either M.2 drive.

◆ While the Linux image is likely to work on other hardware, no guarantee is made as to its suitability, reliability or performance on such hardware.

See the PIXILAB Wiki for instructions on how to obtain and install the Linux image.

In addition to a tuned Linux server core, the Linux server hardware/image also comes with the following features preinstalled and configured:

- PIXILAB Blocks Server software and associated additional runtime components and configuration.
- Server redundancy support, based on the Blocks mirroring service (requires license add-on).
- SSH for secure remote access.
- DNS and DHCP server functionality.
- Remote management software.
- Development tools and libraries for device drivers and advanced scripting.
- Network troubleshooting and diagnostics software.
- Artnet/ DMX-512 testing software.
- Windows/Mac networking interoperability (SMB).
- Windows/Mac USB memory stick interoperability (using the ExFAT file system).

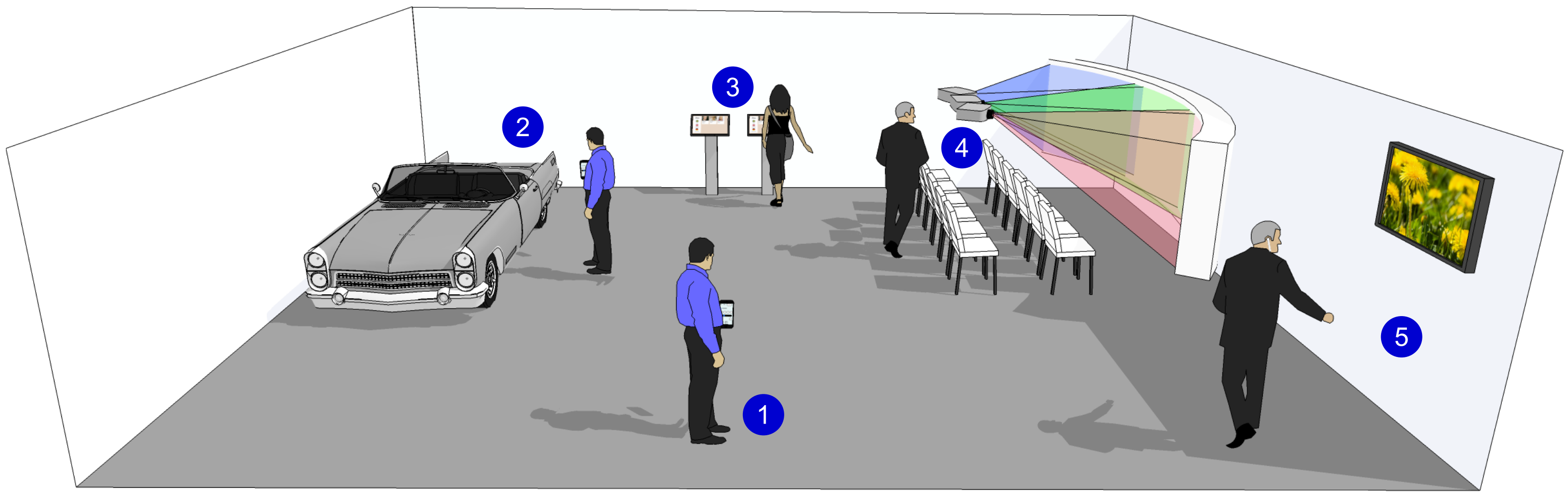
All of the features listed above are pre-installed and configured, but most auxiliary functions are disabled in the default installation for security reasons. Instructions on how to enable them, as well as associated security implications, are found under "Linux-based Server" at https://pixilab.se/docs/blocks.

3. CONCEPTS

This chapter gives a brief introduction to concepts used in Blocks, such as spots, blocks and tasks. These concepts are then explored more in detail in the following chapters.

Spots

In the most general sense, a spot is a location. This may be a geographic position, a room, or an object inside a building. In the context of Blocks, a spot is often related to some media, such as a sound or a video, relevant at that location. While a spot may simply be a location, it is frequently also associated with some equipment, such as a display, touch kiosk or a theater.



Spots inside a building.

Referring to the illustration above, here are the spot types you'll encounter in Blocks:

1. A *Visitor spot* in the form of a visitor carrying a personal, mobile device.
2. A *Location* associated with an object or a position, in this case the car.
3. A touch kiosk based on a *Display Spot* showing an interactive block.
4. A theater with a wide screen show, where a *WATCHOUT Spot* provides synchronized audio through visitors' mobile devices.
5. A regular Display Spot, playing video, images or other content produced in Blocks. Again, synchronized audio in multiple languages can be provided through visitors' mobile devices.

See the chapter titled "Spots" for full details on the various kinds of spots.

Blocks

In Blocks, the objects you use to build content are (somewhat confusingly) called *blocks*. Just like LEGO bricks, those blocks can be combined in a wide variety of ways, allowing you to build different kinds of presentations. While spots relate to physical locations or displays, blocks are more abstract.

Although some blocks can be used on their own (for example, you can use just a *Text block* to show a text message), blocks are commonly combined in clever ways. This is done by putting blocks inside other blocks – somewhat like those Russian Matryoshka dolls.

Here are some other block types:

- *Timeline* and *Schedule* for managing control functions and content playback based on time.

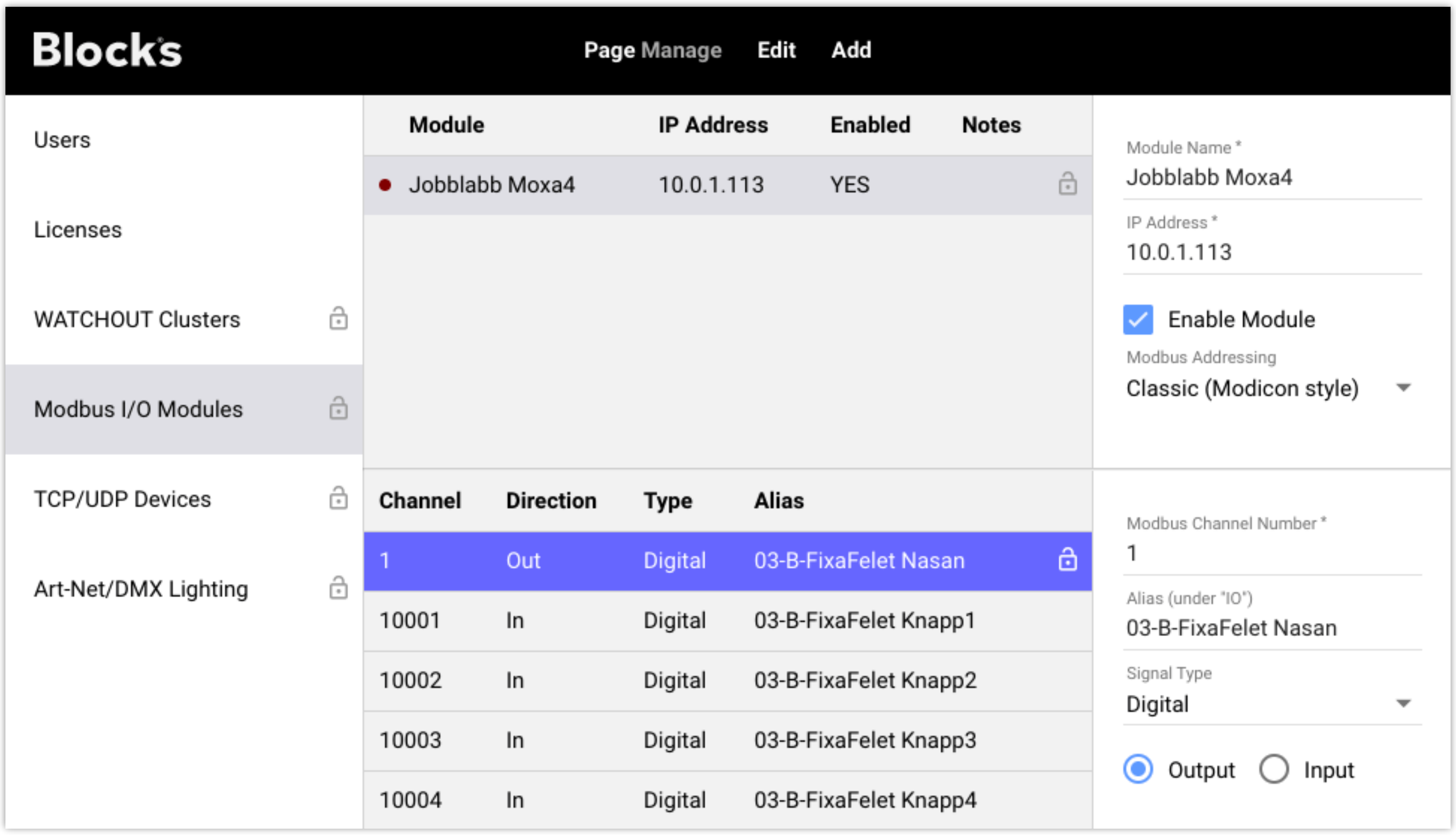
- *Slideshow* for showing a sequence of images, video or other content.
- *Book* for creating interactive presentations using multiple pages.
- *Composition* for arranging other blocks or interactive elements onto a page.
- *Scroller* for interactively scrolling content horizontally or vertically.
- *Attractor* to display an attract loop until a visitor interacts with the spot, then switching to its active content.
- *Synchronizer*, synchronizing playback of video and/or audio across spots. For instance, synchronizing audio played through a mobile phone with video shown on a display.

Buttons, sliders and other interactive controls are also used in a manner similar to blocks. However, such controls can only be used inside a *Composition*.

The chapter titled "Blocks" describes each type in detail.

Managed Subsystems

While *spots* and *blocks* take care of content production, display and interactivity, PIXILAB Blocks can control and manage many other things. Those things are added and configured on the Manage page.



Module	IP Address	Enabled	Notes
● Jobblabb Moxa4	10.0.1.113	YES	

Module Name *
Jobblabb Moxa4

IP Address *
10.0.1.113

✓ Enable Module

Modbus Addressing
Classic (Modicon style)

Channel	Direction	Type	Alias
1	Out	Digital	03-B-FixaFelet Nasan
10001	In	Digital	03-B-FixaFelet Knapp1
10002	In	Digital	03-B-FixaFelet Knapp2
10003	In	Digital	03-B-FixaFelet Knapp3
10004	In	Digital	03-B-FixaFelet Knapp4

Modbus Channel Number *
1

Alias (under "IO")
03-B-FixaFelet Nasan

Signal Type
Digital

[✓ Output] Input

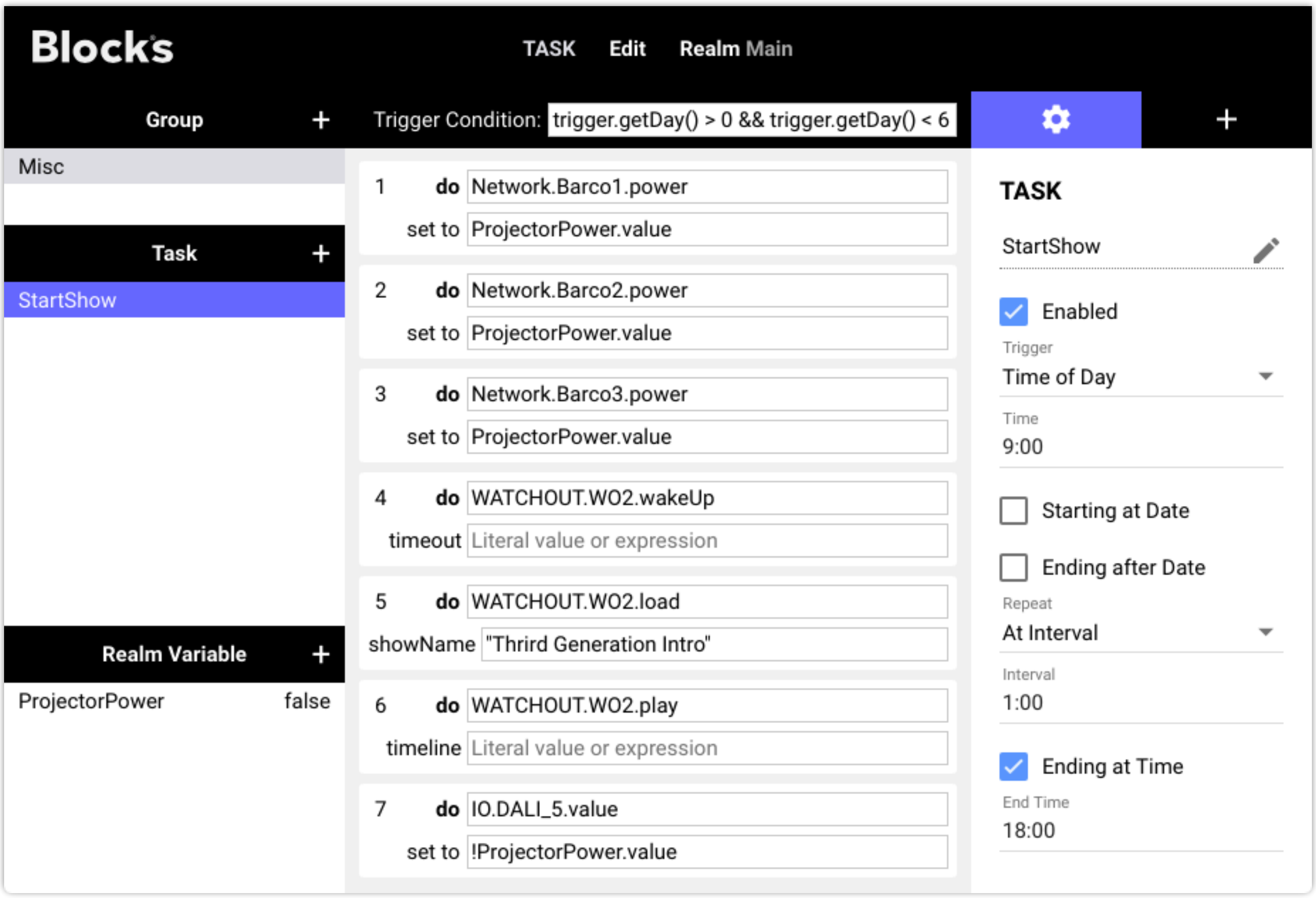
The image above shows how to add Modbus-based I/O modules, for controlling and interfacing with contact closures and sensors. Modbus is an industry standard protocol supported by a large number of vendors. In the same way, you can manage lighting (through Art-net/DMX512), network devices such as projectors (turning power on/off, switching inputs), external data sources (e.g., a barcode reader), etc. Specialized software, called device drivers, are available for interfacing with such network devices. WATCHOUT clusters can also be controlled, for integrating large-scale projection and display capabilities into Blocks.

Tasks

Many basic control functions can be accomplished simply by linking buttons directly to the things being controlled. This kind of one-to-one linking is done directly in the Composition block editor, as you add and position the buttons. As a bonus, this method also provides live feedback of device status, as the status is indicated by the button. All this without any “coding” on your part.

However, some functions are not as simple as a one-to-one relation between a button and a device function. Furthermore, direct linking is just that – direct. There are no *if*s, *but*s or other choices. Sometimes you want things to happen only under certain conditions; such as not on weekends, or not unless there are more than 10 persons in a room. Other times, you may want *different* things to happen depending on some condition.

All this, and much more, can be handled on the Task page. This page in Blocks provides a basic, but surprisingly powerful, programming environment. Just like you combine content blocks on the Display page, the Task page lets you chain together programming statements using drag and drop. Pick the properties to control on drop-down menus, just as you do when linking a button. Then use a simple expression language to determine the details of what to do and how. This expression language is based on JavaScript; the *lingua franca* of the web.



ProjectorPower false

Trigger Condition: trigger.getDay() > 0 && trigger.getDay() < 6

1	do	Network.Barco1.power
	set to	ProjectorPower.value
2	do	Network.Barco2.power
	set to	ProjectorPower.value
3	do	Network.Barco3.power
	set to	ProjectorPower.value
4	do	WATCHOUT.WO2.wakeUp
	timeout	Literal value or expression
5	do	WATCHOUT.WO2.load
	showName	"Thrird Generation Intro"
6	do	WATCHOUT.WO2.play
	timeline	Literal value or expression
7	do	IO.DALI_5.value
	set to	!ProjectorPower.value

TASK

StartShow

[✓] Enabled

Trigger: Time of Day

Time: 9:00

[ ] Starting at Date

[ ] Ending after Date

Repeat: At Interval

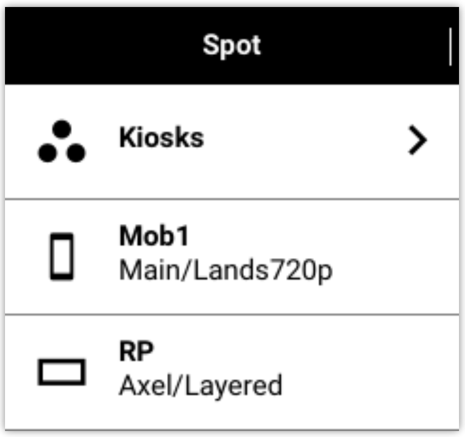
Interval: 1:00

[✓] Ending at Time

End Time: 18:00

4. SPOTS

This chapter provides detailed descriptions of the various spot types. See “Spots” in the Concept chapter for an introduction. You find all your spots in the list shown on the left hand side on the Display page. Spots may be nested inside a Spot Group, in which case you need to click the arrow symbol to enter the group.



The leftmost column of the Spot list contains a summary of the spot’s properties. Drag the right edge of the Spot column to see more details. Doing so reveals two more columns:

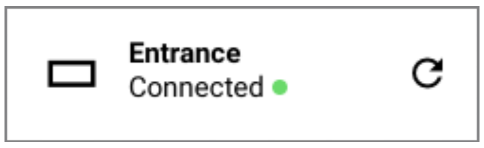
1. **Assigned Block** states the type and name of the block currently assigned to the spot.
2. **Playing Block** states the block currently playing (grayed out if the spot is off line). This is often the same as the assigned block, but may be different if the assigned block is a Schedule, or if the assigned block has been overridden by a priority block. This can be done by a button or by a task.

If the assigned or playing block symbol shows a question mark, this indicates that the block is not found, and therefore can not be played. This can happen if you set a non-existing block with a button or a task, or if you have removed/renamed the block after assigning it to the spot.

To make a Spot play a particular Block, simply drag the Block onto the Spot on the Display page (see “Displaying a Block”). If you drag a block to a Spot Group, the block will play on all spots in that group unless overridden for a nested spot.

Spot Status

The connection status of spots are indicated in the list. If a spot is on-line, a reload symbol is shown. Furthermore, when widening the Spot column to see more details, a “Connected” indicator is shown, or – for a Visitor spot – the number of current connections.



After making changes to a block, click the reload symbol to show the revised block (or enable “Auto Update” in the spot’s settings). Shift-clicking the reload symbol also reloads the display spot’s software.

When using PIXILAB Player with its Offline Spot feature enabled, the reload symbol will spin, and any Connected indicator will be orange, for as long as the default block is being cached. Do not unplug the player while caching is in progress if you then want to use the player offline.

Spot Settings

Double-click the Spot to change its settings. Settings vary with the type of spot, but often include Name, Tags, Location ID and-Role assignments.

Spot Name

Used to identify the spot in the user interface as well as for controlling it using buttons and tasks.

◆ Avoid changing the name of a Spot if you have programmed any actions associated with that spot in buttons or tasks, as doing so breaks the connection.

Tags

You can assign any set of *tags* to a spot. A tag is an arbitrary word or combination of words. Tags are separated by a comma. Tags can be used in conjunction with a Tag Selector block to select the most appropriate content to play on that spot. Additional tags can be applied to a spot using a Button (mainly used for Visitor spots).

Location ID

Associates a short, numeric ID with the spot. This is used in conjunction with a Locator to track down the spot by entering its ID on a numeric keypad. This can be used to create a mobile guide, where visitors play relevant content by entering a number shown on a sign. You only need to assign a Location ID to spots you want to use in this manner. Often, a two or three digit number is sufficient.

Role Assignments

You can limit who can access or control a spot based on role:

- **Role Required to Use Spot.** Limits who can use and operate this spot. This is particularly useful for staff operator panels, and such, providing system control functions that should not be accessible to unauthorized visitors. When set to anything but "None", log-in will be required before providing access to the spot.
- **Role Required to Control.** Limits who can control properties of this spot, such as its assigned block or its power on/off state.

Display ▭

Represents a single display, typically in a fixed location. As Display Spot, you can use;

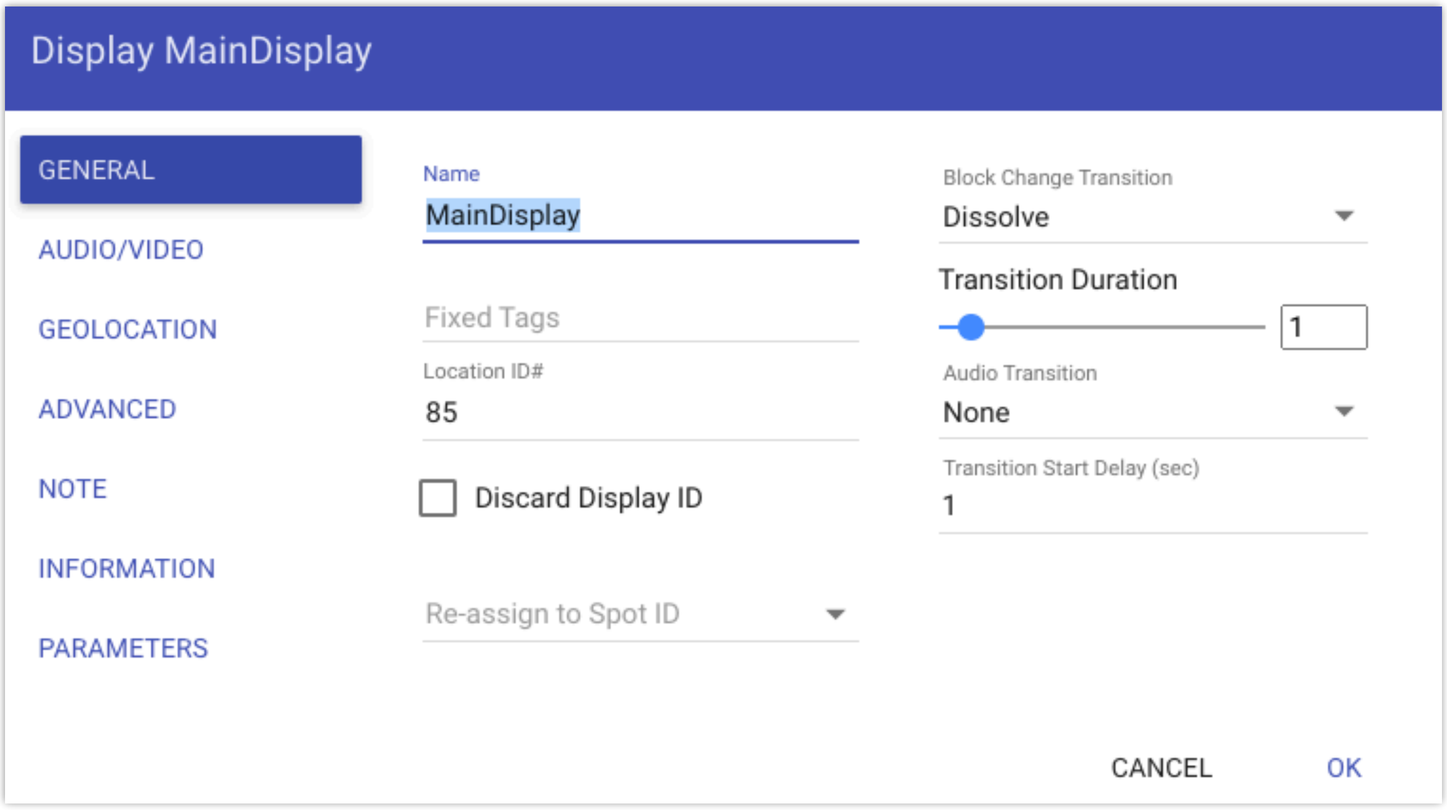
- A regular web browser running on a laptop.
- A small PC, such a NUC, running our free PIXILAB Player software.
- A "smart display," with built-in player available from vendors such as Samsung, LG, Philips, Panasonic or EIZO.
- Other signage players with full support for HTML5 content, such as some Brightsign player models.

Such displays/players typically connect to the Blocks server using a wired network. In addition to Name, Tags and Location ID (see above), it includes the following settings. A Blocks server is licensed for a maximum number of Display and Visitor spots. A message will be displayed to this effect if you try to add more display spots, in which case you'll need to contact PIXILAB to expand your license.

◆ **IMPORTANT:** Performance and capabilities vary widely across players. For best performance and functionality, use our PIXILAB Player software running on an adequately powerful NUC, or equivalent. When using smart displays or signage players, make sure to test all functionality using representative content before choosing such devices for use with Blocks. Some Blocks features, requiring modern browser capabilities, may not work on some smart displays.

General

Basic settings, shared with other spot types, as described above.


Display MainDisplay

GENERAL
AUDIO/VIDEO
GEOLOCATION
ADVANCED
NOTE
INFORMATION
PARAMETERS

Name: MainDisplay

Fixed Tags

Location ID#: 85

☐ Discard Display ID

Re-assign to Spot ID

Block Change Transition: Dissolve

Transition Duration: 1

Audio Transition: None

Transition Start Delay (sec): 1

CANCEL OK

Discard Display ID

This checkbox appears only for a spot that has already been assigned to a display. It's useful if you want to detach the player, in order to re-assign that player to another Display Spot.

Re-assign to Spot ID

This option appears only when there are new, unassigned displays connected. It allows you to re-seat this Display spot to one of the unassigned ones. This is useful in cases where you want to replace a physical player with another player, while keeping all settings intact. Depending on the type and model of player being assigned, some settings may no longer apply. For example, some settings found under Video/Audio may no longer apply if the new player doesn't have the same hardware configuration as the old one.

Small Blocks

Just like displays, most block types also have pre-determined dimensions. If the block is smaller than the display, this setting determines what will happen:

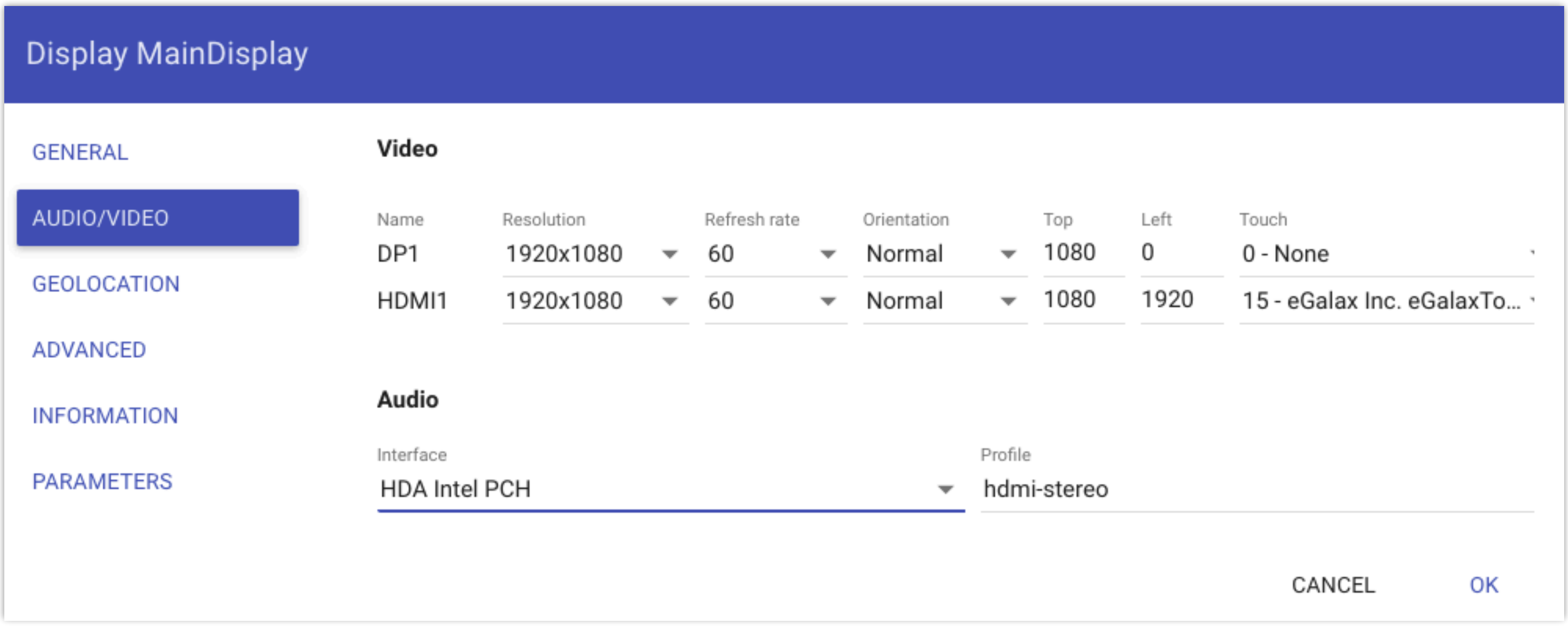
- **Scale to Fit** scales the block (while maintaining its aspect ratio) to fill the display.
- **Center** the block in the middle of the display, without scaling it.
- **Top Left** puts the block in the corner of the display. Useful when used with an LED wall processor or similar, in which case it's preferable to always put the block in a known position, such as the top left corner.

Block Change Transition

Selects a transition to be applied when changing the block playing on the spot. This is particularly useful when the playing block will change due to user interaction or a Schedule. See "Transition" in the Slideshow block for details on available transitions. The "Transition Start Delay" option gives the incoming block some time to load its content before appearing on screen, if required.

Video/Audio

The fields found under this heading vary with the type of player being used. Basic settings include the width and height of the display, as specified when added to Blocks. When used with PIXILAB Player version 3 or later, the following options are provided.



Video

Name	Resolution	Refresh rate	Orientation	Top	Left	Touch
DP1	1920x1080	60	Normal	1080	0	0 - None
HDMI1	1920x1080	60	Normal	1080	1920	15 - eGalax Inc. eGalaxTo...

Audio

Interface: HDA Intel PCH
Profile: hdmi-stereo

Name, Resolution, Refresh Rate and Display Orientation

Controls the resolution, refresh rate and display orientation of each display connected to this Display spot.

Note that the *Name* field may not be representative of the physical connection used. For instance, in the illustration above, the outputs are named DP1 and HDMI1, even though they are both HDMI connectors.

- ◆ While you can have multiple displays connected to one player, the player still represents a single Display Spot. Any Block assigned to this Display Spots will span the total, combined pixel space of all its displays.

Top, Left

These columns appear only for players with multiple outputs, in which case each display's relative position and other settings can be specified independently to arrange displays horizontally, vertically or in a grid. You can position displays to compensate for the width of display frames or specify overlapping image areas for projectors with built-in edge blend functionality.

Touch

If the display has a touch overlay, this is typically connected to the player using USB. Make sure to select the correct touch device here for each such display. The *Touch* menu may list other input devices, not related to touch. It may also list the same touch input multiple times with slightly different names, so some experimentation may be required to get this right.

- ◆ **NOTE:** The default "0–None" setting may or may not provide reliable touch input, depending on the operating system's default behavior. Avoid this setting when using a touch overlay, even if it may appear to work at first glance.

Audio Settings

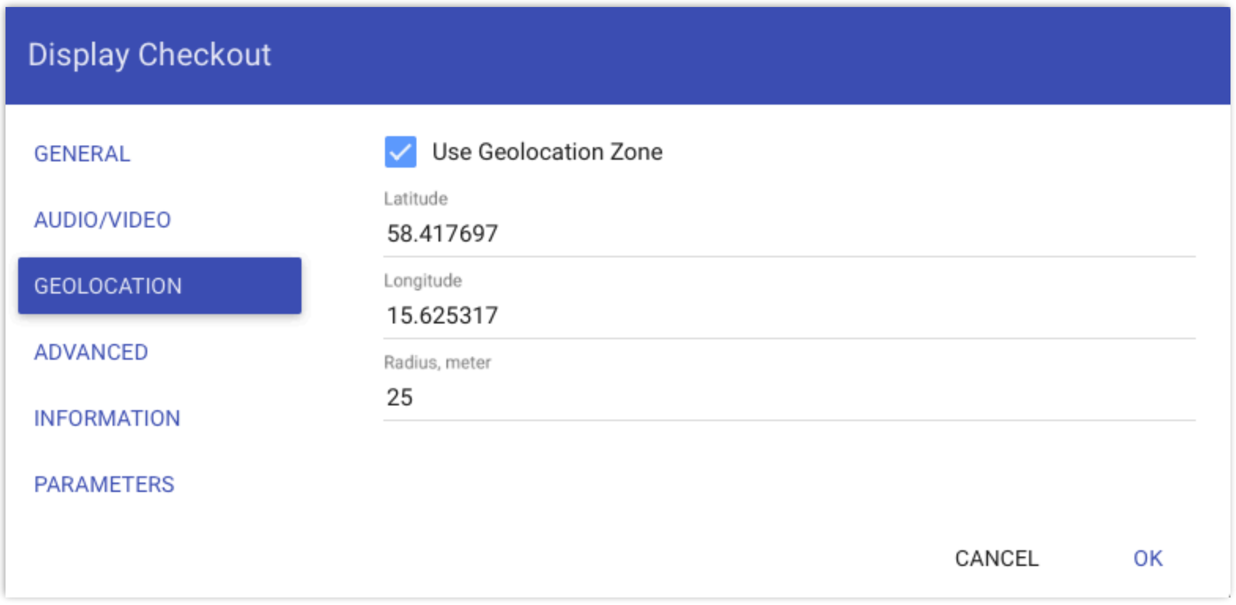
Selects the audio output device and configuration. This is useful if there are multiple sound output options available, such as analog and HDMI, to ensure that the desired output is used. Make sure to select the correct audio output here. It may also list the same output multiple times with slightly different names, so some experimentation may be required.

- ◆ **NOTE:** The default "Auto" setting may or may provide reliable audio, depending on the operating system's default behavior. Avoid using this setting even if it may appear to work OK.

Initial Volume specifies the volume level applied when the spot connects to the server, normalized as 0…1. The **Volume Fade Rate** setting results in a smooth fade rather than a jump when setting the spot's *volume* property.

Geolocation

Assigns a geographical position to the spot, which can subsequently be used in conjunction with the GPS functionality of the Locator. This is similar to the Location ID mentioned above, in that it allows you to track down a location, but uses your GPS position rather than having to enter a number to indicate your location.



Use Geolocation Zone: ✓
Latitude: 58.417697
Longitude: 15.625317
Radius, meter: 25

Specifying a geolocation zone by latitude, longitude and radius.

Advanced

Various more rarely used options, including authorization requirements for the Display spot.

Sync Offset

Lets you compensate for differences in the propagation delay inherent in some display devices. Generally not required if using displays of the same kind or if displays aren't acting in a synchronized manner. if required, set this to a small positive or negative value, in milliseconds, to account for any display propagation delay differences.

Update PIXILAB Player

Select the desired version here to update PIXILAB Player to that version. This feature is only available when:

- The player runs PIXILAB Player software version 7 or later.
- The player starts from an internal SSD (i.e., not from a USB stick).
- The current user has the role of Manager or Admin.
- Another version of PIXILAB Player software is found on your Blocks server.

Player updates must be stored here inside Blocks' root directory on your server:

```
public/PIXILAB_Player/updates/pixilab-player-XXX.img
```

where XXX is the version number. The current version of PIXILAB Player can be obtained from here. Updates must be named exactly as shown above, so don't rename them after download.

Auto-update on Block Change

The block assigned to this spot will be reloaded automatically when modified in the editor. Enable this option to get live feedback while editing blocks. In general, don't enable this for spots in public view, as that would make the spot reload frequently whenever its block is edited. Instead, use the manual reload button once you finish editing the block to publish the revised block.

Caching
In some cases, content can be stored on the player rather than constantly streamed from the server. This reduces the network traffic, and can allow some content to be played even without a server connection.

- **Disabled** prevents content from being cached on the player even if the player and your license supports this.
- **Enabled** activates general caching of video and other content if supported by your license and player, thereby reducing network traffic. Supported by PIXILAB Player version 6 with persistence enabled and PIXILAB Player 7 or later when installed on the internal SSD. It may also work with other players if there's at least 5GB of storage and the server connection uses HTTPS (HTTPS not required with PIXILAB Player).
- **For Offline Use** caches the Spot's default block (the one dragged onto the Spot on the Display page) and all its associated media files on the player. This allows the block to be played even when the player is not connected to the server. Such offline use does not support any functions relying on the server.
    - ◆ **IMPORTANT:** Offline playback works only with our PIXILAB Player software. The player needs to support persistent storage, such as an internal SSD (SDXC cards are *not* compatible). A separate "Offline Spot" license is required for each player to be used offline. This license remains consumed even if that player is no longer connected to your system or you later disable the Offline mode. Mark such players clearly. Players for which Offline use has been enabled are tracked in a file named "Licenses-XXX" in the model directory of your Blocks root directory, where XXX is your license number. Do not rename, delete or share this file across servers or Blocks root directories.

Clear Cache
Resets all data stored on the player. This can occasionally be useful if you believe that the data stored on the player may have become out of sync or corrupt for some reason. Available only for PIXILAB Player 7.1 and later.

Preferred Synchronization Master
This option exists only on Display Spots inside a Spot Group. Selecting this option designates the Display Spot as the master when playing a Synchronizer block on spots in this group (e.g., by dragging the block to the Spot Group). If none is selected, Blocks will pick a master automatically, so in most cases you don't need to specify this.

You may want to specify the master explicitly if you at some point need to borrow a displays from this group for other purposes. Then designate a spot that will *not* be borrowed as the master to keep the still-synchronized spots working properly. Do not select more than one as Preferred Synchronization Master in a Spot group. When viewing the Spot list in expanded mode, this setting is indicated by the word "sync" next to the online indicator.

Power Down Automatically
If selected, and the Display spot has provisions for controlling power, it will be turned off when no block is assigned to the spot. This is particularly useful together with a "Schedule" as it will then manage power on/off automatically accordingly, turning power off whenever no block is being shown.

Blank Schedule Inherits Group
If a Schedule block assigned to this spot yields nothing, then play the block determined by the enclosing group, which may in its turn use a Schedule or other method for setting the block. This option exists only on Display Spots and Spot Groups inside a Spot Group.

Numeric Keys as Inputs
When enabled, the numeric keys in the main keyboard area can be used as GPIO inputs. These inputs are then exposed as a number of read-only, *keyPress* properties of the spot. This is particularly useful in conjunction with various keyboard-emulating devices.

- ◆ When selected, this option will interfere with normal use of a keyboard to enter text and numbers. This option is mutually exclusive with "RFID/QR Scanner Input" described below.

RFID/QR Keyboard Input

When enable, keyboard input will be exposed as the *scannerInput* property of the spot. This is useful in conjunction with one of the following:

- A handheld player that has a built-in QR/barcode scanner, such as this Zebra device.
- An RFID/NFC reader connected to the USB port of the spot (supports "keyboard emulation" readers only).
- A Locator block set to "Use RFID/QR Scanner" to establish the Location ID of a spot (as an alternative to using the numeric keypad.
  - ◆ When selected, this option will interfere with normal use of a keyboard to enter text and numbers. This option is mutually exclusive with "Numeric Keys as Inputs" described above.

RFID/QR Input Lingers

Available only when "RFID/QR Scanner Input" is selected. Defines how long (in seconds) the scanned code remains set on the property. This setting is useful for scanners that repeat the code as long as the RFID/NFC tag is nearby. Set this to a time value slightly larger than the reader's repeat interval. This has two benefits:

- It avoids setting (and clearing) the property repeatedly, potentially re-triggering any action caused by the tag.
- It allows detecting also when the tag *is removed*, by the fact that the *scannerInput* property then becomes false (empty string).

Keyboard keyCode Property

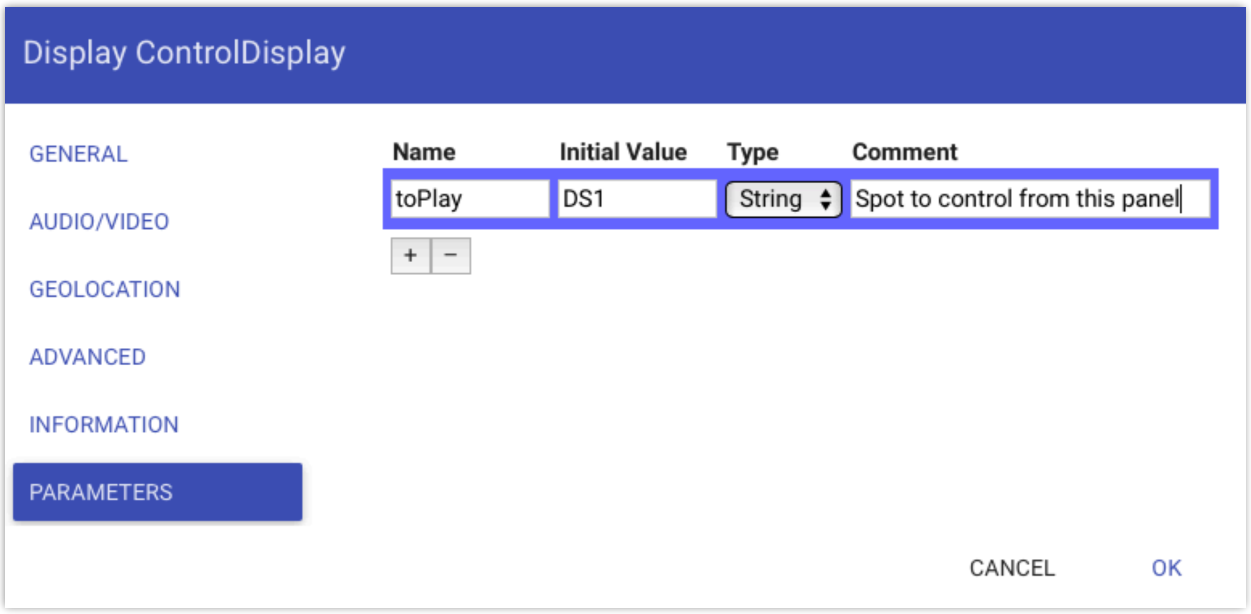
When enabled, a *keyCode* property will be provided by the Display Spot, indicating the physical key code of the last key pressed on its keyboard.This is particularly useful as a general purpose input signal from a connected keyboard-emulating device. Not available when "RFID/QR Scanner Input" is selected.

Note, Information and Screenshot

- **Note** lets you add arbitrary notes or comments to the spot.
- **Information** shows information about the spot and player – particularly useful with PIXILAB Player.
- **Screenshot** shows what's on the Display Spot's screen now (requires PIXILAB Player version 7.1 or later).

Parameters

Assigns initial values to *spot parameters,* specific to this spot. This allows you to define or override the initial value of specified parameters on a spot-by-spot basis. See "Spot Parameters" for details on how to use this feature.



Name	Initial Value	Type	Comment
toPlay	DS1	String	Spot to control from this panel

Spot URL Query Parameters

The display spot player or web browser uses a URL as described earlier in this manual (see "Adding a Display"). Following the mandatory part of the URL, you can list a number of optional *URL query parameters*. Each such parameter consists of a name and a value, separated by an equal sign. The list of URL parameters begins with a question mark. Subsequent parameters are separated by an ampersand. The following parameters are supported:

- **display=*N*** allows multiple display spots to run in separate browser windows, where each window has its own number N.
- **mobile=*SPOT*** indicates that this is a Visitor Spot, where *SPOT* is the name (dot-separated path if in a group) of the corresponding Visitor Spot in Blocks.
- **mac=*MAC*** uses the specified *MAC* address as the unique identifier for the display spot instead of a dynamically allocated ID. Useful on players with no local storage to hold the unique ID.
- **spot-id=*123XYZ*** Apply 123XYZ a custom identifier on a spot, rather than relying on the automatic ID assignment. Occasionally useful if you want to lock a player to a specific ID and the automatic ID assignment can't be used for some reason.
- **class=*CLASSNAMES*** adds the *CLASSNAMES* (comma separated if more than one) to the display root element. Allows the same block(s) to appear with different CSS styling on different spots.
- **tags=*tag1,tag2,tag3*** applies the specified set of tags to the spot. This is similar to setting a fixed set of tags in the general display settings, but allows you to specify those as part of the URL.
- **param-*XXX*=*value*** sets the initial value of the spot parameter named *XXX* to *value*. This is similar to assigning parameters in the settings dialog, as described above, but allows you to specify those as part of the URL.
- **spot=*group.name*** or ***Location ID*** makes any active Locator locate the specified spot using its name (with group and spot names dot-separated) or the Location ID of a spot. This is particularly useful in conjunction with the QR code or NFC modes of the Locator, with the same code specifying both the Visitor Spot's URL and your location.

For example, to use an additional display spot managed by a separate browser window plus as a custom class name:

```
http://pixi.guide/spot?display=2&class=myspecialclass
```

Visitor ▯

Represents a mobile device, such as a phone, iPad or Android tablet. Such devices can be used for a wide variety of applications:

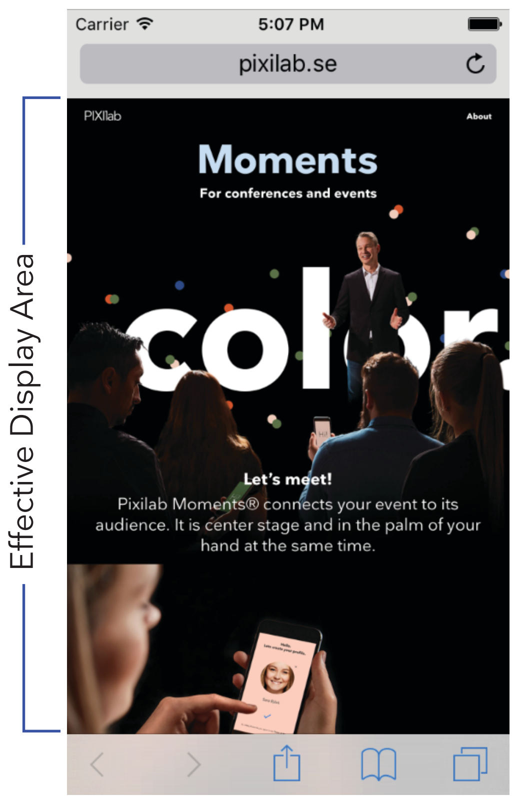
- Visitors' phones (assuming the Blocks server is accessible, e.g., through a local, wifi network).
- Loaner units, for use by visitors not bringing their own devices.
- Locally managed devices, used as operator panels by the staff.

Visitor Spot Dimensions

Unlike regular Display spots, where you know the exact dimensions of the display when you add the spot, a Visitor spot may be accessed using a variety of devices, with varying screen sizes and orientations. This is especially true for bring-your-own devices. Hence, when adding a Visitor spot, you often need to consider some least common denominator in terms of display size.

While using a larger or taller display is generally not a problem (except that buttons and other controls may look oversized), assuming a taller display than what some visitors have may result in some elements appearing off screen. Therefore it's usually a good idea to be a bit conservative here. For instance, you may decide that the display of the iPhone SE is a good least common denominator to aim for. While the display resolution is 1136 pixels tall by 640 pixels wide, some pixels are lost to the address bar at the top and the navigation buttons at the bottom, resulting in about 920 by 640 pixels of effective display area.

Note that the exact pixel values you specify aren't that important, as you're essentially specifying an *aspect ratio* and coordinate space for laying out content. While it make things easier for you if the sizes of content and display match, the content will eventually be scaled to fit the width of whatever device it being used.


Carrier
5:07 PM
pixilab.se
PIXlab
About
Moments
For conferences and events
color
Let's meet!
Pixilab Moments® connects your event to its audience. It is center stage and in the palm of your hand at the same time.
Effective Display Area

If you want to make the height more dynamic, use a Stack Block and set the main content to "Fill Remainder".

Accessing a Visitor Spot

Unlike a Display spot, where there's a one-to-one correspondence between a physical display and the item in the Spot list, there may be any number of physical devices associated with a single Visitor spot. All mobile devices that connect to the URL of this spot will initially show the same content. After adding a Visitor spot to the Spot list, access it with a URL like this:

```
http://10.2.0.10:8080/spot/?mobile=Mob1
```

This is similar to the URL used for a Display Spot, as shown under "Adding a Display". Replace 10.0.2.10 with the IP address of your server. The number ":8080" after the IP address is the port number used by the server. This may also be different depending on your server's configuration. If you're using the standard http port ":80", you don't need to specify this part at all in the URL.

◆ The reason the Blocks server is pre-configured to not use the standard port 80 is that some operating systems don't allow applications to use ports below 1024 without authorization. Windows does not have this restriction, making it easy to switch to port 80 here if desired. But for the sake of consistency, the default port also under Windows is 8080. When using our Linux server, the standard port 80 can be used, as an internal redirection is in place.

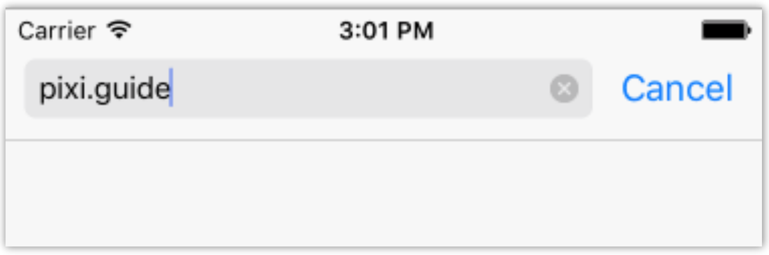
The last part, beginning with the question mark, indicates that this is a Visitor spot. The name entered after the equal sign is the name of the Visitor spot (here "Mob1"). If the Visitor spot is inside a Spot Group, you need to prefix this name with the name of the enclosing group(s). For instance, if it inside a group called "Mobiles", you would use an URL like this:

```
http://192.168.0.33:8080/spot/?mobile=Mobiles.Mob1
```

While this kind of configuration is OK for locally managed devices and loaner units, it's not very practical for visitors. To simplify visitor on-boarding, you need to change some server options:

- Make the server use port to 80.
- Set the server's default redirect to "/spot/?mobile=Mob1" (or whatever is the path to the desired Visitor spot).

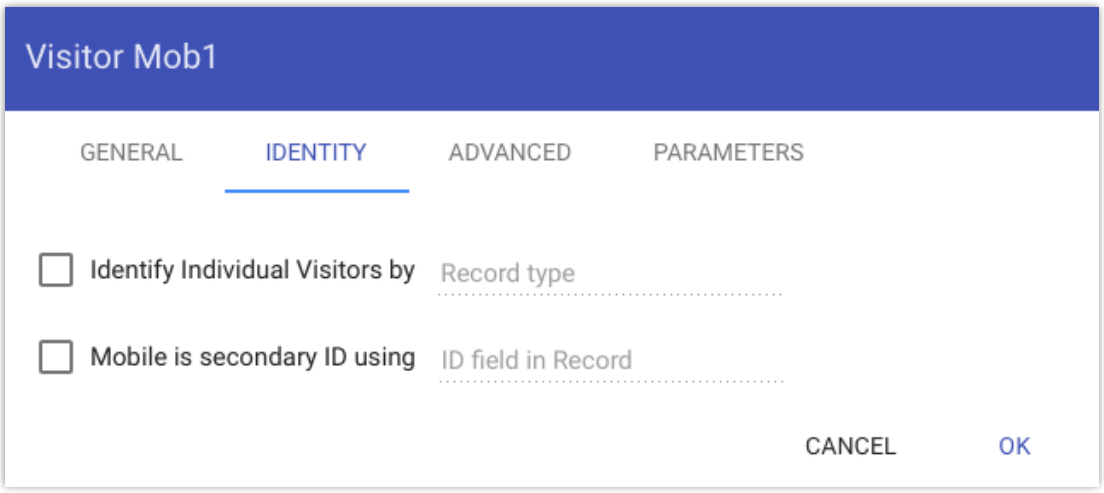
Both these options are described under "Server Configuration Options" in the Blocks wiki. Once this has been done, entering just the server's IP address into the browser is sufficient. To further simplify the process, you can use a DNS server that associates an easy-to-remember name with the server's IP address. This DNS is used by the visitor-accessible network. Assuming you've pointed the name "pixi.guide" to the server's IP address (10.2.0.10 in the example above), you can simply type *pixi.guide* into the address field of your mobile phone's browser to access the guide functions, as shown below.



◆ Our Linux-based server includes all you need to set this up as described above, including the DNS server functionality.

Visitor Identity

The Identity tab in the Visitor Spot's settings lets you identify and interact with individual visitors.


Visitor Mob1

GENERAL IDENTITY ADVANCED PARAMETERS

☐ Identify Individual Visitors by Record type

☐ Mobile is secondary ID using ID field in Record

CANCEL OK

◆ **IMPORTANT:** These settings take effect only when Visitor Data Collection is activated in your license.

Identify Individual Visitors by

Select this option and enter the name of a valid *Record Type* to enable the tracking of, and interaction with, individual visitors. This option provides the ability to control and interact with each person through his/her mobile device. A *User Script* must be provided, defining the *Record Type* specifying what kind of data to collect. This *User Script* also controls how to interact with the visitor or what happens during the visit.

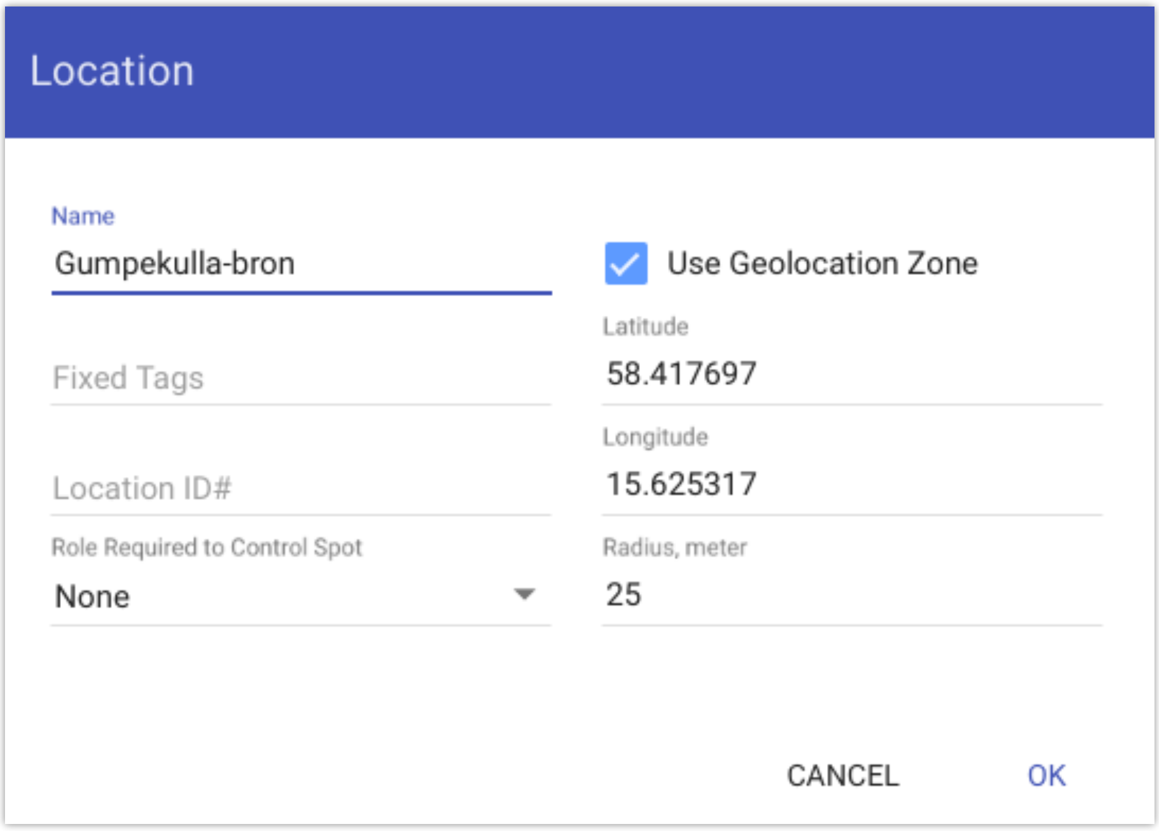
Mobile is Secondary ID using

Select this option if the mobile device accessing the Visitor Spot isn't the primary means of identifying the visitor. You may, for example, prefer to use an RFID tag or other ID token as the primary means of identification, as to not require that each visitor has a mobile device. However, if a visitor does have a mobile device, you may still want to use that to further enhance and personalize the visit. If so, select this option, then enter the name of the *ID* field in your *Record Type* used to hold the identity of the mobile device. You User Script is responsible for assigning the mobile's identity (referred to as its *PUID* inside the script) to that field in order for Blocks to then associate both the ID token and the mobile device with the same data record.

Location

Represents a physical location, or an object of interest (such as the car in the illustration under "Spots" in the Concepts chapter). This is useful when you want to associate content with this location, to be accessed using visitors' mobile devices. This may be a sound clip for an audio guide, a video, a textual description or anything else you want to present at this location.

Add a Location to the spot list (you may first want to create a Spot Group for this purpose). Give it a name, and assign it a Location ID. This ID must be communicated to the visitor, often using some kind of clearly marked sign at that location. Associate the content you want to appear on the visitor's mobile device with this spot. Then, using a Locator, the visitor enters the *Location ID* of the spot to access this content (such as *44* shown below for a WATCHOUT Spot).



Location

Name
Gumpekulla-bron

Fixed Tags

Location ID#

Role Required to Control Spot
None

✓ Use Geolocation Zone

Latitude
58.417697

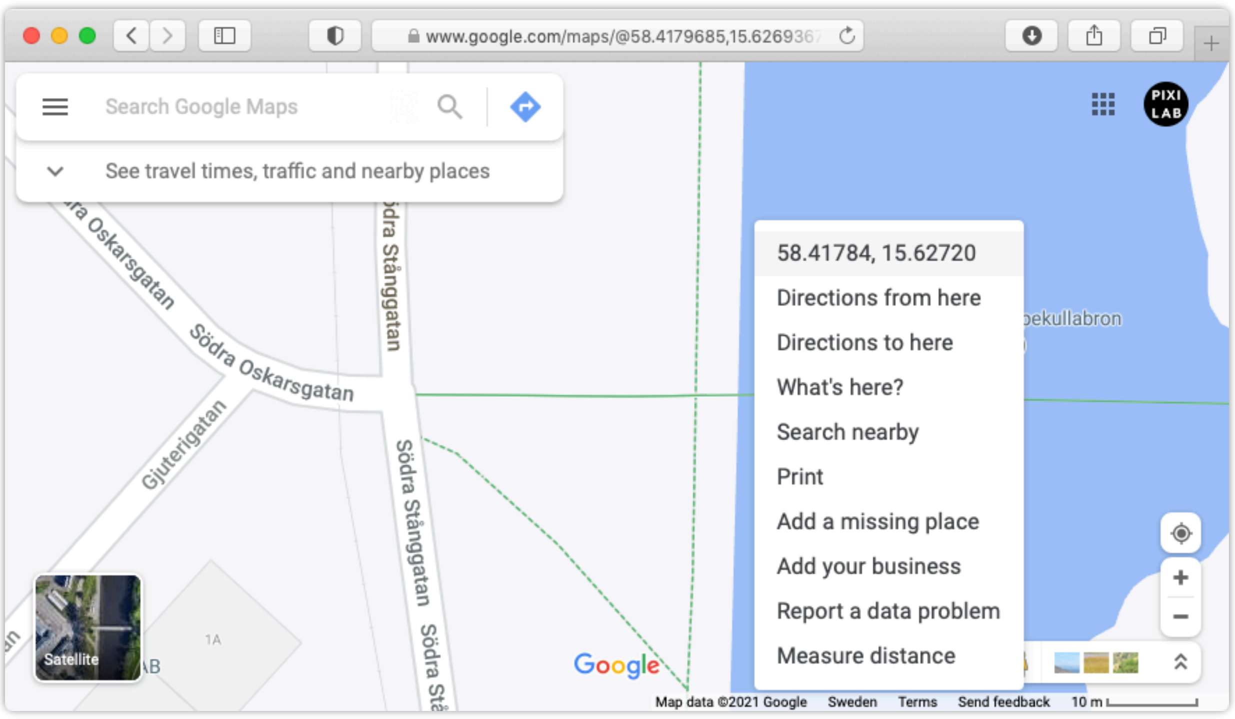
Longitude
15.625317

Radius, meter
25

CANCEL OK

Use Geolocation Zone

Select this option and enter the location and radius to use the GPS feature of the Locator to locate this spot. This provides an alternative method for positioning, based on your GPS position rather than entering a location ID number on a numeric keypad. You can obtain the geolocation using most map applications, such as [Google maps](), and map applications of mobile phones. Specify a sizable radius around the location to allow for some error in the GPS data.



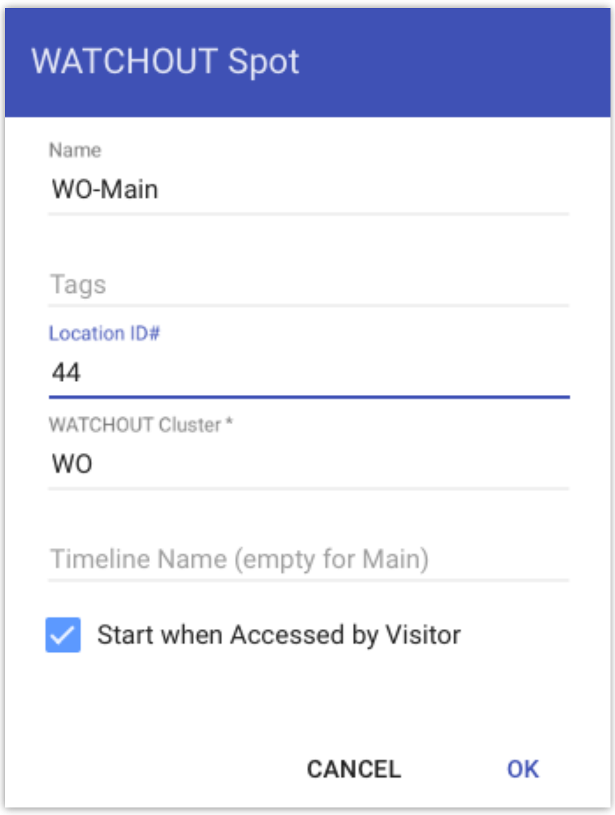
Right-click in Google maps to pick the GPS position.

◆ **NOTE:** Geolocation based on GPS works reliably only outdoors. Buildings and other large structures may degrade the accuracy significantly.

Overlapping Zones

Geolocation zones may overlap. When they do, Blocks will chose the smallest matching zone. Thus, when creating a geolocation based experience, you may want to create a very large zone, enclosing all your actual points of interest. This allows you to provide some content to appear outside those locations, such as a map showing where the points of interest are or some other message explaining how to find them.

WATCHOUT Spot


WATCHOUT Spot

Name
WO-Main

Tags

Location ID#
44

WATCHOUT Cluster *
WO

Timeline Name (empty for Main)

☑ Start when Accessed by Visitor

A WATCHOUT spot allows you to play audio synchronized with a presentation running on Dataton WATCHOUT, which is sometimes used in theaters or other locations with specialized requirements, such as projection onto a curved screen or some physical object (projection mapping). While WATCHOUT can also play the sound track associated with such a presentation, you may want to provide narration in additional languages through visitors' mobile devices.

◆ Unlike Display and Visitor spots, a block associated with a WATCHOUT Spot will *not* appear on the WATCHOUT screen. Instead, the block will appear only on Visitor spots when navigating to the location of the WATCHOUT spot.

In addition to adding a WATCHOUT spot, you need to add and configure a WATCHOUT Cluster on the Manage page. You must also obtain a sound track matching the target timeline of the WATCHOUT show. The sound track must match the timeline starting from its beginning, even if the content on the timeline may not start at the beginning. For instance, if the content starts five seconds into the WATCHOUT timeline, you must add five seconds of silence at the beginning of the sound clip to match up.

◆ To playable on mobile devices, the sound file must be encoded using the AAC format, typically stored in an .m4a file.

WATCHOUT Cluster
Enter the name of the cluster here, as specified on the Manage page.

Timeline Name
Enter the name of the WATCHOUT timeline to synchronize to. Leave this field empty to synchronize to the main timeline.

Start when Accessed by Visitor
If checked, Blocks will tell WATCHOUT to start playback when a visitor accesses the spot by entering its Location ID into a mobile device. Use this method to start the WATCHOUT show "on demand" by the first visitor. Subsequent visitors will join at the current time position, and will not restart the show.

If the WATCHOUT show is started by other means (e.g., by a scheduled Task), leave this unchecked.

Spot Group

A Spot Group acts as a "folder", allowing you to group spots in ways that make sense for your installation. You may use a group merely for organizational purposes, representing a floor or a room in the building. Spot Groups can be nested inside each other, allowing you to have an outer group for the floors, with inner groups for the rooms on each floor.

You can also use a Spot Group to associate related displays. For instance, you may have a set of three displays performing together. Put those into a Spot Group, and then assign content to all three in one go by dragging the Block to the Spot Group rather than to each individual spot inside. Using tags assigned to the spots inside the group, in combination with a Tag Selector or Synchronizer block, you can target the content to the matching display (e.g., tag the displays left, center and right, and use a Tag Selector configured accordingly).

Finally, you can use a Spot Group to synchronize video playback across a number of Display spots:

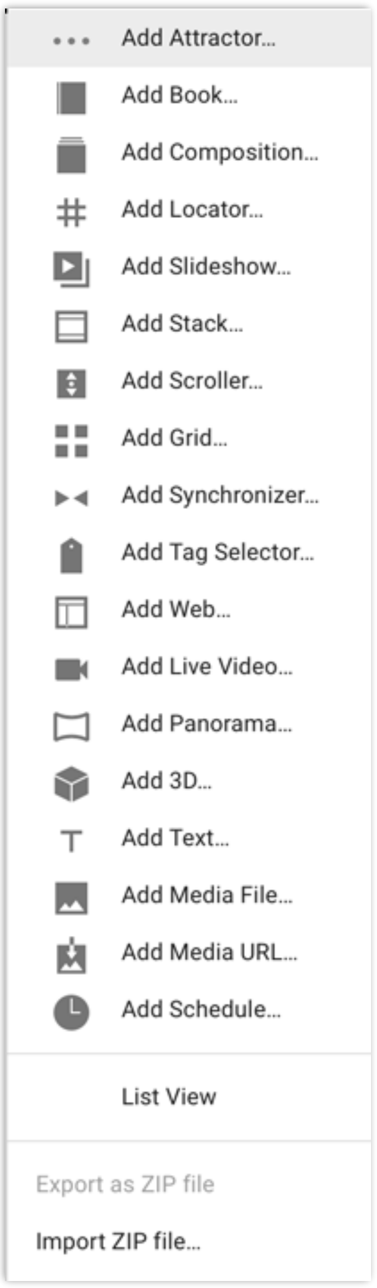
- Place those Display spots in a Spot Group.
- Tag each display as appropriate (e.g., *left*, *center* and *right*).

- Use a Synchronizer to play the videos, with corresponding tags assigned to each of the videos to make them play at the desired display.
- Assign that Synchronizer to the Spot group (which will make it apply to all spots in the group unless overridden).
- Optionally, designate one of the displays in the group as the Preferred Synchronizer Master in Group.

See also under “Synchronizing Video Across Multiple Displays”.

5. BLOCKS

Content is managed using various types of blocks. While some block types may seem simple, their true power becomes apparent when combined in creative ways. See under Blocks in the concept chapter for a brief introduction.



A block can be either at the top level of the Display page, or nested inside another block. Top level blocks can be manually assigned to spots by dragging the block to a spot on the Display page. They can also be dynamically assigned using buttons and tasks. Alternatively, use a Schedule to change the block assignment based on time, date or weekday. When the block assignment changes, the corresponding displays are updated accordingly.

Creating Blocks

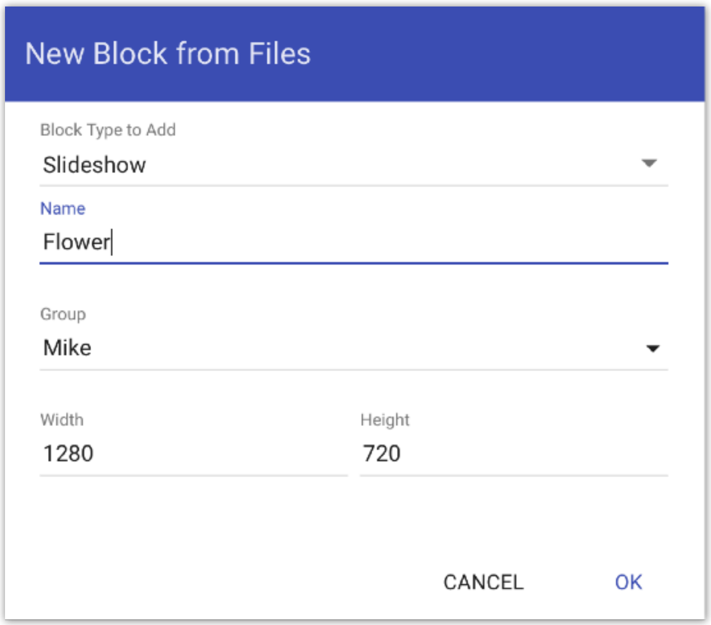
Create a top level block using the Display page's Block menu, as shown to the right. Many block types can contain other blocks. For instance, a Slideshow often contains images and video, but can also contain more complex block types, such as a Composition or a Web block.

Some of the most commonly used block types are:

- **Composition** for combining multiple blocks, buttons or other controls on one page.
- **Book** for arranging content across several pages.
- **Slideshow** for showing a sequence of images or other blocks, with transitions.
- **Timeline** for time-accurate control and content playback functions.

Creating a Block from Media Files

As an alternative to the manual method described above, you can create some common block types from a set of media files. For instance; to create a slideshow from a set of images, just drop those images on the background of the Display page, chose "Slideshow" in the dialog box that appears, specify name, group and other settings and click OK.


New Block from Files

Block Type to Add: Slideshow
Name: Flower
Group: Mike
Width: 1280
Height: 720

Block Group
When creating a top level block, you give it a name and associate it with a *block group*. Initially, there's a single group called "Main". To establish a new group, type the desired name into the Group field when adding the block. Organizing blocks into groups makes them easier to find as the number of blocks increases. Planning ahead, deciding up front how to name and group blocks, will save you time later when using those blocks.

Once you have lots of blocks, it may be hard to find the blocks you're working on. If blocks are organized into groups, you can filter the set of blocks shown on the Display page using the Group menu. Here you can choose to see just a single group, or blocks from all groups.

List View
Select "List View" on the Block menu to view the top level blocks in a more compact way. Sort the list on any column by clicking its title.

Hiding Blocks Based on Role
By associating groups and individual blocks with *roles*, you can hide blocks for all users that don't have the designated role (or higher). Hiding blocks in this way can make the system easier to use and less confusing. See "Authorization and Roles" for details.

To limit the visibility of all blocks in a group based on role:

1. Select the desired group on the Group menu of the main Display page.
2. Click the padlock symbol next to the group name.
3. Choose the required role in the dialog box.

To set the visibility of an individual blocks:

1. Double-click the block on the Display page, to view its editor.
2. Choose the role on the "Role Required to See" drop-down menu.

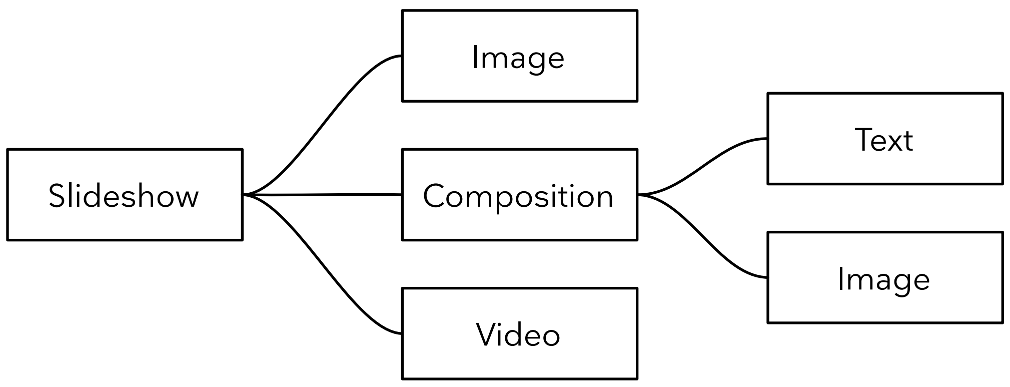
If you set the role at the group level, this role is inherited by all blocks in that group that don't explicitly override this setting.

- ◆ To see the effect, you must be logged in as a user with a role below the one specified for the block's visibility.

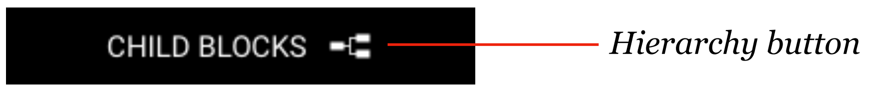
Editing Blocks

Edit a block by double-clicking it on the Display page. Click the Blocks logotype in the top left corner to return to the Display page, or choose Display on the Page menu.

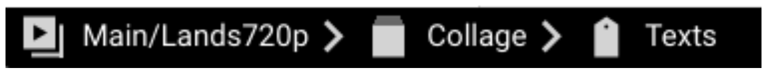
Blocks can contain other blocks, in a tree-like fashion. For instance, you may have a block hierarchy that looks like this:



View the block hierarchy using the hierarchy button at the top of the Child Blocks list. The hierarchy view also lets you navigate directly to any block by double-clicking it.



A block that contains nested blocks, such as the Composition in the illustration above, is opened by double-clicking it in the list of Child Blocks. While inside such a nested block, “breadcrumbs” are shown along the bottom of the window. Clicking the name of an outer block here returns to that level.



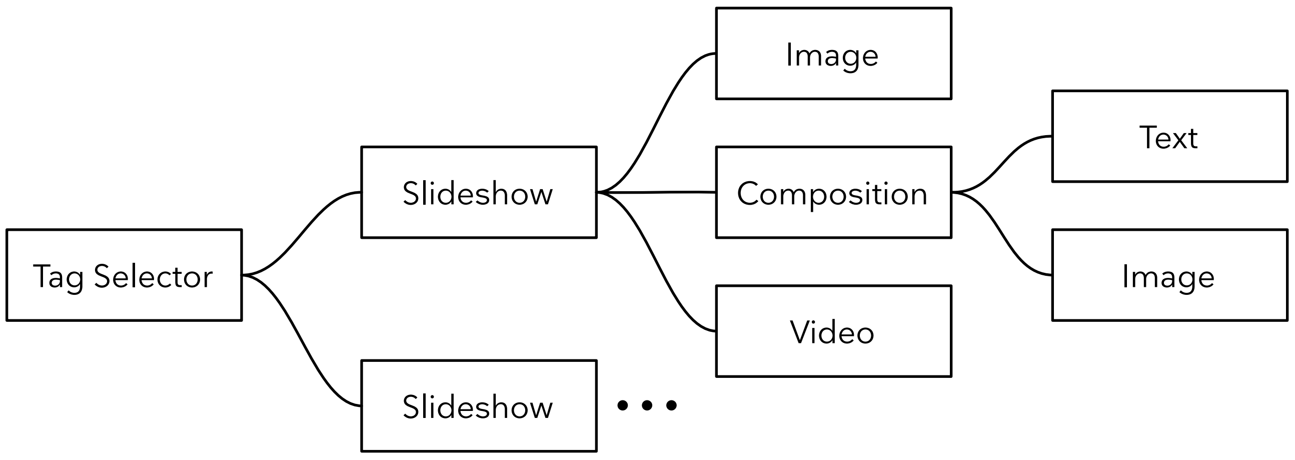
Breadcrumbs show types and names of enclosing blocks, allowing you to navigate back to any outer level.

Adding Child Blocks

Add a child block either by dragging media files into the Child Blocks list, or through the “Add Child” menu. The various block types are described later in this chapter. For images, video and audio, see under “Media File” in this chapter.

Wrapping a Root Block

The “Wrap With” menu appears only at the topmost level of a block. This menu allows you to insert a new block at the topmost (root) level, moving the previous root level block into the added block. For instance, starting with the above block hierarchy with a Slideshow as the outermost block, you may later find that you need a Tag Selector as the outermost block, in order to present two different slideshows depending on some tag (e.g., to support multiple languages). To do so open the Slideshow editor, then chose *Wrap With > Tag Selector* to insert a Tag Selector as the root level block, moving the Slideshow one level down:



Wrapping the Slideshow root block with a Tag Selector. This allows you to subsequently add other blocks, such as a second, alternative slideshow.

Previewing a Block while Editing

When editing a complex block with many nesting levels, it’s sometimes useful to see the result in context while editing. You can do so by one of the following methods:

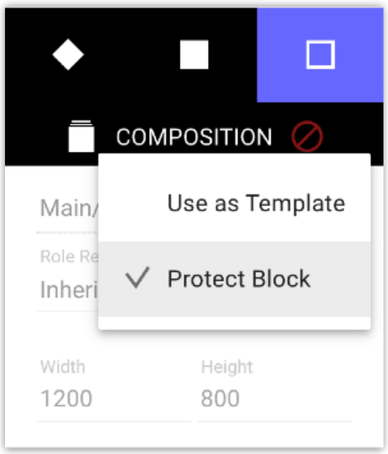
- Before you begin editing, drag the block to a Display Spot you use as a preview monitor. Select “Update on Block Change" under the Advanced tab in the Display Spot settings. This allows you to use a real spot (such as a PIXILAB Player) to preview the result. As you change settings in the edited block, the Display Spot will auto-navigate to reveal the edited block.
- Chose “Show Editor Preview" on the Edit menu while editing a block. This opens a separate window on your computer, showing the result in context. Note that this preview window isn't a complete spot, so this method may not work property if your block depends on Spot Parameters or other spot-specific features. If so, use the previous method to preview the block while editing.

Exporting and Importing Blocks

Use the commands at the bottom of the Block menu to export and import blocks, either for archival purposes or for transferring to another Blocks system. To export a block, first select it then choose “Export as ZIP file”. This downloads a ZIP file with the name of the block to your computer. Such a ZIP file can later be imported using the “Import ZIP file” command. Choose a ZIP file containing a previously exported block then specify the desired group and name for the imported block.

Block Protection and Templates

You can protect a block or its content from accidental modification, or mark a block as a template used as a basis for creating new blocks. These features are selected on the *protection menu*, accessed through the symbol shown at the top right of the settings panel, next to the block’s type name. The color and shape of the symbol indicate the current protection. Click the symbol to pop up the menu, as shown on the right. Depending on the context, this menu contains a combination of the following:



- **Use as Template** is available at the root block level only, and marks the block as a template (see below).
- **Protect Block** is available for all blocks at all levels. If selected, the block can’t be changed. Select this option for blocks that shouldn't be deleted, moved or changed. Note that unless you also select “Protect Block Content”, you may still change any child blocks inside.
- **Protect Block Content** is available for Text blocks and blocks that can contain child blocks, such as compositions. If selected, you will not be able to edit the text or descend into the block by double-clicking it, thus protecting its content.

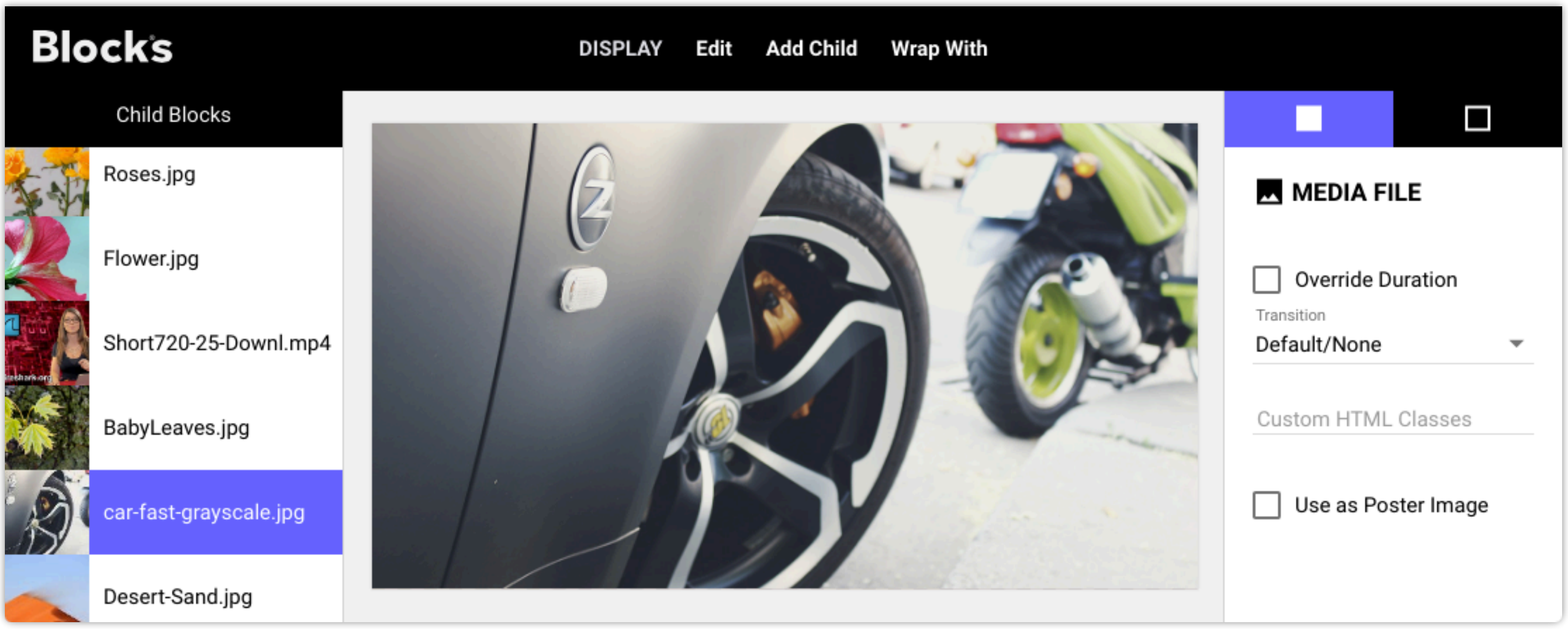
◆ **NOTE:** Selecting “Use as Template” or “Protect Block” for a root block also prevents this block from being deleted from the Library page. This is in addition to any such protection provided by the current user’s role.

Using Templates

If you often create blocks that are alike, you can save time by first creating a block with all the basic content in place. Then mark this block for “Use as Template”. When you later attempt to edit this block, you’ll be prompted to make a duplicate first, using a different block name. That copy will then be opened for editing, keeping the template original unchanged. Users with the role of Creator or higher can chose to edit the template or first make a copy of the template. Users below the role of Creator can not edit the template, but must always make and edit a copy.

Slideshow

A Slideshow presents its child blocks sequentially, with a transition between each one.



Child Blocks

The list along the left side of the editor window shows the child blocks that make up the slideshow. Drag blocks in this list to change their order of appearance. Double-click a complex block to open its editor (does not apply to simple blocks, such as images).

Click a child block in the list, or the solid square in the top right hand corner, to view the settings of the selected child block. Settings here vary with the type of child block selected, as well as with the type of the outer block.

Override Duration

By default, a slide is shown for the duration specified for the slideshow in its Outer Block Settings (see below). Video is shown for its entire duration. Click "Override Duration" to change the duration for this particular child block, then set the desired duration by dragging the slider or typing into the field. While the slider only goes to 10 seconds, you can type any duration into the field.

Transition

When set to "Default/None", the transition specified for the slideshow in its Outer Block Settings will be used (see below). Select another type to apply that transition for this block only.

Custom CSS Classes

Applies a CSS class (or several space-separated classes) to this block, allowing it to be styled in a particular way when used in conjunction with custom CSS. See Custom Styling for more details.

Use as Poster Image

Selects the child block to use to represent this block on the Display page. By default, this is taken from the first visible child block. You may pick a more representative child block using this checkbox. This setting has no effect on the playback of the block.

Outer Block Settings

Click the hollow square in the top right hand corner to edit the settings for the slideshow as a whole. Many of the settings found here are also available for other block types. Click the pen symbol next to the block name to rename the block, or to move it to another Block Group.

- ◆ Renaming or moving a block that's in use will break references to this block (e.g., Spot assignments and Reference Blocks), and you will have to re-establish those manually.

Role Required to See

This option appears only at the top level, and allows you to limit who can see this block on the Display page to a certain role or above.

Width/Height

The dimensions of this block. You set those dimensions when first creating the block. The settings can be changed here. In general, you don't change this after creating a block. If you do so, you should revisit the places where this block is being used, to make sure it appears as expected. Typically, the block dimensions should match the resolution of the Spot on which it will be displayed, or at least have the same aspect ratio.

Custom CSS URL and Custom CSS Classes

These fields allow for custom styling of buttons and other elements.

◆ **NOTE:** CSS styling is an advanced feature. See Custom Styling for more details.

The "Custom CSS URL" field applies only to top-level blocks, and may be one of the following:

- A full URL, beginning with "http://" or "https://", pointing to any reachable web server.
- An absolute Blocks server URL, beginning with "/public/", for loading a CSS file stored under the *public* directory on your Blocks server
- A block-local URL, beginning with "~/" (such as *~/style.css*), for loading a CSS file stored inside the blocks's folder on the server. This is useful for CSS that applies to a single block only. Any such CSS file stored in the block's folder will be included when a block is exported/imported.

◆ Create or edit a block-local CSS file, by clicking the pen icon in the "Custom CSS URL" field. This creates a block-local CSS file if one doesn't already exist, and opens an editor window for that file. This feature can also be accessed through the "Edit Block CSS" command on the Edit menu, which is available also while editing a child block.

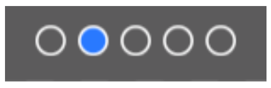
Finally, it is also possible to automatically load custom CSS globally for *all* Spots in the system using the defaultSpotCSS option in the server configuration file.

Play Automatically and Loop

In most cases, you want a slide show to start playing immediately when shown. If the slide show is a top level block, you typically also want it to loop. If a slideshow is used as an inner block (e.g., inside an outer slideshow), you likely want the inner slideshow to play only once, then allowing the outer block to proceed.

Show Slide Position

Indicates the total number of slides and the current slide graphically, using a number of circles at the bottom of each slide, with a filled circle corresponding to the current slide.



◆ **HINT:** Use CSS to style the look of these symbols. As an alternative to this graphical representation, there's also a textual indication provided in the form ***X/N*** where *N* is the total number of slides and *X* is the 1-based index of the current slide. This type of indication is provided when "Show Slide Position" is not enabled, but is not shown by default. It can be shown, styled and positioned, using a few lines of CSS, like this:

```
.page-counter {
    display: block;  /* Makes the text appear */
    right: 0.8em;    /* Positions the text a bit from the right edge */
    color: white;    /* Text color */
    text-shadow: 0 0 0.3em black; /* Makes text visible against bright background */
}
```

Pause Video when Slideshow Paused
If selected, the current slide will be paused when the slide show is paused. This is sometimes useful for slides containing video. If this option is not selected, the slideshow will be paused, but any video will continue to run to its end, and pause there, waiting for the slide show to resume playback.

Pause without Stopping
If selected, you can pause the slideshow without triggering any enclosing Attractor set to close when playback ends. If selected, an additional setting appears controlling how long the slideshow may sit in this paused state before considering it stopped (thereby triggering any enclosing Attractor).

Default Slide Duration
Determines how long to show child blocks that have no inherent duration, such as still images. A video or other child block that has an inherent duration will be shown in its entirety (unless you override the duration of that child block).

Default Transition
The transition used between child blocks in this slide show, unless overridden for a child block. See Transition below for details.

Child Replication
Allows slides to be generated automatically based on data from an external database or similar. This is done by replicating a single, prototypical child block. See Child Replication under the Book block below for more details.

Transition

The following transition types are available:

- **Cut.** No transition. The new image (or other child block) appears instantly.
- **Dissolve.** The new image gradually replaces the previous.
- **Crossfade.** The new image appears gradually up while the previous fades away. Useful to as an alternative to Dissolve when transitioning between images with different aspect ratio or where one of time images contain transparent areas.
- **Fade through Color.** A black or colored solid covers the current image, and then fades out to reveal the next image.
- **Slide In.** The new image slides in over the top of the old one. You can then also specify the direction.
- **Push.** The new image slides in as the old one slides out.
- **Pop.** The new image jumps out over the old one from behind.
- **Zoom.** The old image fades out and scales down/up as the new appears, with an optional "spin" effect.
- **Wipe.** The new image appears by moving an edge across the previous image. Both images remain stationary.
- **Custom (CSS)**. Lets you create custom transitions and animations based on custom CSS applied to the block or globally. See examples below.

Most transitions allow you to specify the duration of the transition (i.e., its speed). Some transitions have additional settings. To see the effect of those settings, change the settings, then click the previous slide in the list and then back to the current one.

Audio Transition
To make the audio transition as well, select one of the following options on the *Audio Transition* dropdown, if available:

- **None.** Any incoming audio starts right away. Outgoing audio is cut off at the end of the transition.
- **Fade In.** Any incoming audio volume ramps up gradually. Outgoing audio is cut off at the end of the transition.
- **Fade Out.** Any incoming audio starts right away. Outgoing audio is faded out.

- **In and Out.** Any incoming audio volume ramps up gradually. Outgoing audio is faded out.

Custom CSS Transition Examples

When selecting the "Custom CSS" transition type you must provide your own CSS implementing the transition. You can also select whether the incoming block appears *on top of* or *below* the outgoing block during the transition. Blocks applies specific classes to the elements, as shown in this example (which replicates the built-in Crossfade transition):

```
.custom-transition-prepare { opacity: 0.001; }
.custom-transition-in { opacity: 1; }
.custom-transition-out { opacity: 0; }
```

To use more than one custom transition type, use the "Custom CSS Classes" field to apply a class acting as a transition type selector and add this to the CSS as an enclosing class for the custom-transition-xxx class selectors. The classes shown above are applied like this:

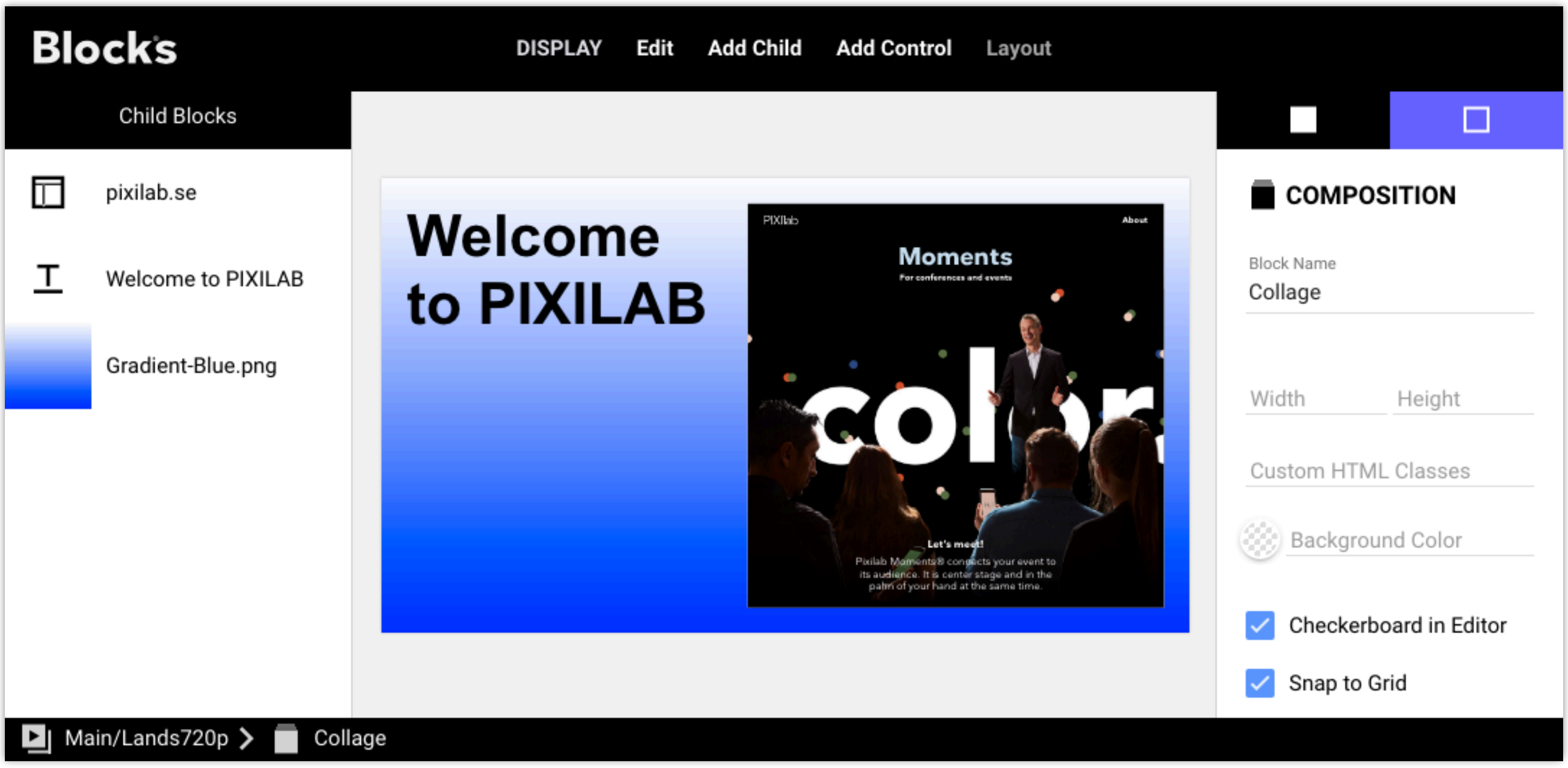
- **custom-transition-prepare** is applied to the incoming block before starting the transition, allowing you to do any initial setup. In the above example, this is used to make the incoming block transparent before starting the fade-up specified by custom-transition-in.
- **custom-transition-in** is applied to the incoming block when starting the transition.
- **custom-transition-out** is applied to the outgoing block when starting the transition.
- **custom-transition-reverse** is applied to both blocks if the transition direction has been reversed by "Auto-reverse Transition" being selected.

In addition to these classes, Blocks also applies *transition-duration* and *animation-duration* as inline styles to both the outgoing and incoming blocks, with the Transition Duration specified.

◆ **HINT:** Here are more custom CSS transition examples.

Composition

A Composition block displays its children together, next to or on top of one another, in a back to front order. In the illustration shown below, the blue gradient is the bottommost layer, with Text and Website blocks on top. Drag child blocks in the list to change their stacking order. The breadcrumbs shown along the bottom indicate that this is a nested block. It's inside the Lands720p Slideshow, which is in the Main block group.



Block Name: Collage
Width
Height
Custom HTML Classes
Background Color
✓ Checkerboard in Editor
✓ Snap to Grid

Many outer block settings are the same as those described above for the Slideshow block.

Name

Give the block a meaningful name to make it easier to find inside its outer block. This name can also be used when navigating to a child block inside a Book, Slideshow, or similar. Thus, avoid changing the block name if you're using it to locate the block from a button, task or similar.

Width/Height

While it is sometimes useful to override the width and/or height of an inner block, in most cases you leave these fields blank to make it inherit the dimensions of the outer block. If this Composition had been a top-level block, those fields are mandatory, and determine the overall size of the block.

Background Color

By default, the background of a composition is transparent. This allows compositions to be used inside outer compositions as a means of grouping inner blocks. If you want the composition to have an opaque background, pick its color here. In the example above, doing so would make no difference as the opaque, blue gradient image fills the entire composition.

Checkerboard in Editor

Specifies the background used when editing the composition. Doesn't apply if a solid background color is selected. Choose a checkerboard style that makes your content clearly visible for editing purposes.

Editor Scale

When using large compositions, the composition is normally scaled automatically to fit within the editor. In some cases, such as when editing text, this may not be desired as it may cause the text to be rendered too small to be edited. Then select the desired scale factor and scroll the page to view the desired part for editing.

Snap to Grid and Grid Size

When selected, elements moved with the mouse will snap to a pixel grid. This often makes it easier to line things up. When selected, you can specify the granularity of motion as the Grid Size. Deselect this option to move items freely. You can always position items numerically, or using commands on the Layout menu, even if this option is selected.

Keep Children Inside

When selected, child blocks can't be dragged outside the enclosing composition. Deselect this option if you want to use the composition to crop its inner block(s), showing only the portion that intersects the composition.

Hide After Delay

When selected, the composition (and all its children) will be hidden after the specified time of inactivity. Touch the display (or move the mouse) to make it reappear. This is useful when interactive controls, such as buttons, are placed on top of other content, thus obscuring the content behind. By checking this option, the controls will be hidden automatically. Just touch the display to make them reappear.

- This option is only available for nested compositions, not for top level compositions.

Prototypical Binding

Allows you to indicate the expected type of binding for use as a target block when navigating from a replicated block. This can, for example, be used to show more details related to the data source associated with the replicated block.

- **NOTE:** This is an advanced option, for use together with child block replication.

When navigating from a nested, replicated block to a block somewhere *outside* that hierarchy, this is needed since the target block then won't have the context of the nested block. Use this option to indicate what type of binding context to be expected by this composition block's children by specifying the same kind of object as you intend to navigate from. Children (such as Text and Media URL) can then use the Relative property root in their bindings.

Child Block Settings

Most child block settings in a composition are type dependent, and are described under each block type in this chapter. The following settings are common for all Composition child blocks.

- **Left, Top, Width, Height.** Position and size of the child block within the composition. Alternatively, drag the child or one of its edges or corners in the main area. To resize an image or video proportionally, press Shift while dragging to resize.
- **Custom CSS Classes.** Modify the look of many child blocks, such as applying custom fonts or styling of buttons and other controls. See Custom Styling for more details.
- **Blend Mode.** Controls how pixels of this child block are combined with pixels from blocks below it within this composition. This is similar to the corresponding function in applications such as Adobe Photoshop, and is occasionally useful for creating effects combining images, video or text – possibly along with behaviors such as Opacity.

Controls and Text blocks can be bound to local or system properties for interactivity. See Controls and Panels for more details.

Layout

This menu provides commands to align selected elements. Select the desired elements either in the child block list on the left or using the mouse. Then choose a command to align the selected items.

Align Horizontally

Moves selected blocks horizontally, making their left, center or right sides line up.

Align Vertically

Moves selected blocks vertically so their top, middle or bottom line up.

◆ If you select a single object, it will be aligned to the enclosing composition.

Set Width Ratio

Set the width of the selected block(s) to the specified ratio of the enclosing block. This makes it easy to size a child block to fill half the width, or some other provided ratio.

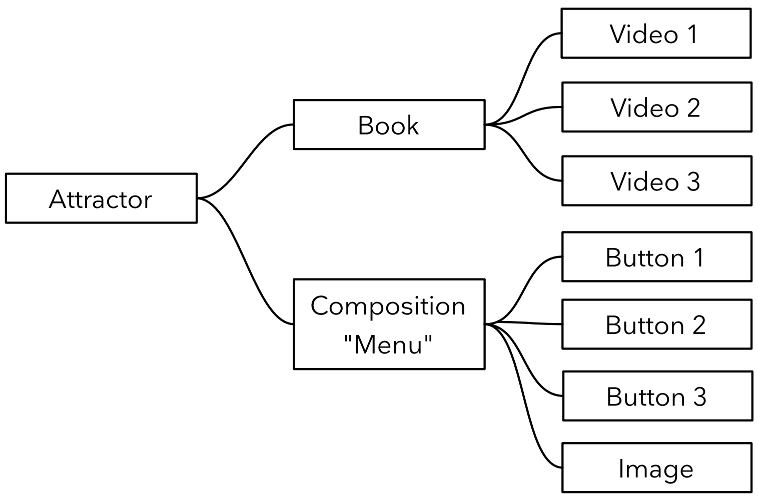
Add Control

Controls include interactive elements such as Button or Slider (see "Controls and Panels" for details). While such elements are mostly treated as other blocks, they only make sense in a Composition since they need to be sized and positioned in relation to each other, and are never used at full screen size. While the Add Child menu is available in all blocks that can have child blocks, the Add Control menu is only available while editing a Composition.

◆ If you want to use controls inside any other block type, first add a Composition, then add the control inside that composition.

Book ▐█

A Book displays its children as separate, named pages. You can navigate to a page using a Task or a button (see "Go to Local Block" under Button). The example below shows a Book with three pages, each one displaying a video. The buttons in the "Menu" Composition control the Book. The Book and the "Menu" Composition is then wrapped in an Attractor, which causes the menu to re-appear once the video shown on a page in the book finishes playing.



- **Page Transition**. Applies a page transition when changing page in the book. See Slideshow above for available transitions.
- **Auto-reverse Transition**. Reverse the direction of some transitions, such as Push and Slide In, when changing to a previous page in the book. Often used with "Swipe to Turn Pages".
- **Swipe to Turn Pages**. Support swipe gestures for user navigation within the book. If not selected, navigation must be specifically catered for using buttons or tasks.
- **Wrap Around at End**. If selected, navigation past the last/first page wraps around to the other end of the book. Applies only when "Swipe to Turn Pages" is selected.
- **Show Page Indicator**. Indicates the total number of pages and the current page graphically, using a number of circles at the bottom, with a filled circle corresponding to the current page. See Show Slide Position under Slide Show for more options.

Other outer block settings are described above under Slideshow and Composition.

Special treatment of first/last page

Occasionally, you may want the first and/or last page in a book to look different than other pages. For instance, you may want to hide or gray out certain elements on those pages. To facilitate this, Blocks adds the CSS class *first-child* or *last-child* to the page's element when at those extremes. You can then use custom CSS to treat those pages differently.

Child Replication

This is an advanced feature, allowing a single child block to be replicated a number of times, with each replicant being bound to a different object. Those objects may originate from indexed sub-properties of a device (such as channels on an audio mixer), or feed items from a data source, such as a museum collection database or other external API. Inside a Book, such replication results in a page for each replicant. Other block types supporting child replication (such as Slideshow, Grid and Scroller) provide other arrangements.

Once you have a device driver with indexed sub-properties, or a feed script for accessing external data, you can use *child replication* to instantiate the corresponding number of child blocks during playback:

- Set the "Child Replication" mode to "Indexed Property".
- Specify a partial binding to the indexed property of the device or feed instance. Any feed instance is found under *Script, feed, <script-name>, <instance-name>*, where *<script-name>* is the name of the feed script, and *<instance-name>* is the name of a feed instance provided by that script.
- Specify how many items to skip from the start of the list (default is zero). This allows you, for example, to show the first mage of a feed in a prominent way, while listing any additional images below, in which case the list of additional images should skip the first one to not show it again.
- Limit the number of child blocks to produce (default is to show all items available from the driver or feed).
- To access feed- or item-specific information in children (such as the current page number, or the total number of pages), enter an arbitrary Replicator Scope name. This can then be used inside replicated children to access such information under *Local, scope, <scope-name>, <property-name>*, where *<scope-name>* is the name specified.
- Inside the replicated child block, relevant properties will appear under *Relative* for binding text, images, controls, etc. Underlined *Relative* properties are directly available for use in replicated child blocks. Non-underlined *Relative* properties should only be accessed from details views, such as when showing a single object – not a grid or list of objects.

  ◆ **NOTE:** In the editor, you'll see only the single, prototypical child block, without any instance-specific data being shown, allowing you to bind text, images, video, etc to the desired properties. The actual replication happens on the Spot only, providing real data to each instance as determined by the data source.

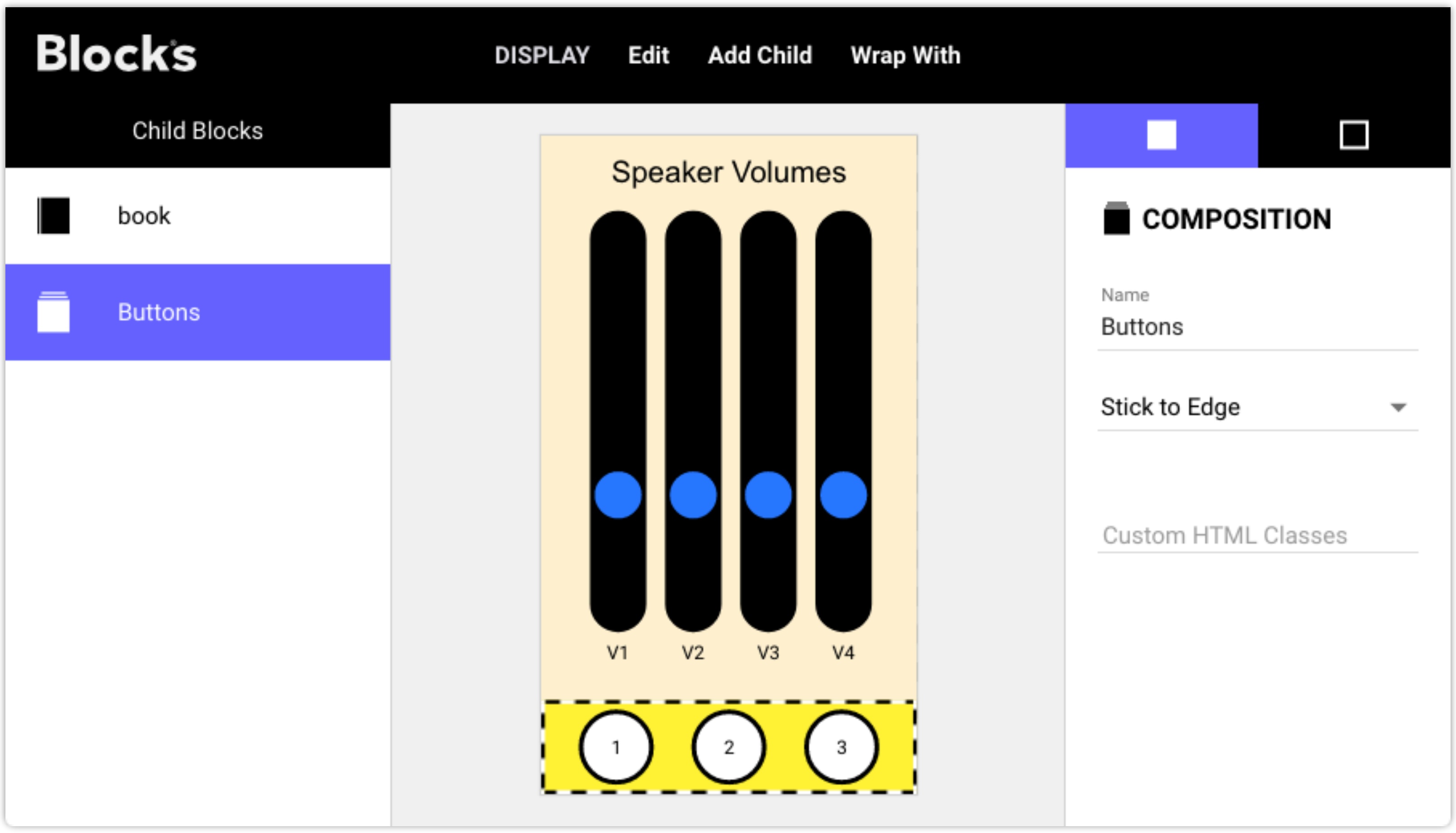
Navigating from a Replicated Child

While the replication mechanism makes all relevant properties available to child blocks, you may sometimes want to bring such property bindings along to a block that is *not* a child block of the replicator. For instance, you may have a Scroller using child replication to list a number of objects, perhaps with a thumbnail image and a title for each. Upon clicking that block, you want to display detailed information about the object. To do so, add a "Go to Local Block" button to go to another block, designed to show the detailed information. However, that *other block* can't in itself be a child of the replicator, since it isn't inside the list.

To facilitate such use, Blocks will pass along the context from the clicked list item to the target block (typically a Composition), allowing it to show data from the originating list item. However, in order to *design* that Composition (which isn't itself inside the replicator), you need to provide the context that will eventually be used so you can bind images, text and other controls at design time. This is done by specifying a Prototypical Binding inside the target Composition.

Stack

A Stack displays its children as two or more vertically arranged sections. It is typically used to add navigation bars along the top and/or bottom of a mobile device or similar, with a larger main area taking up the center of the screen.



Name
Buttons

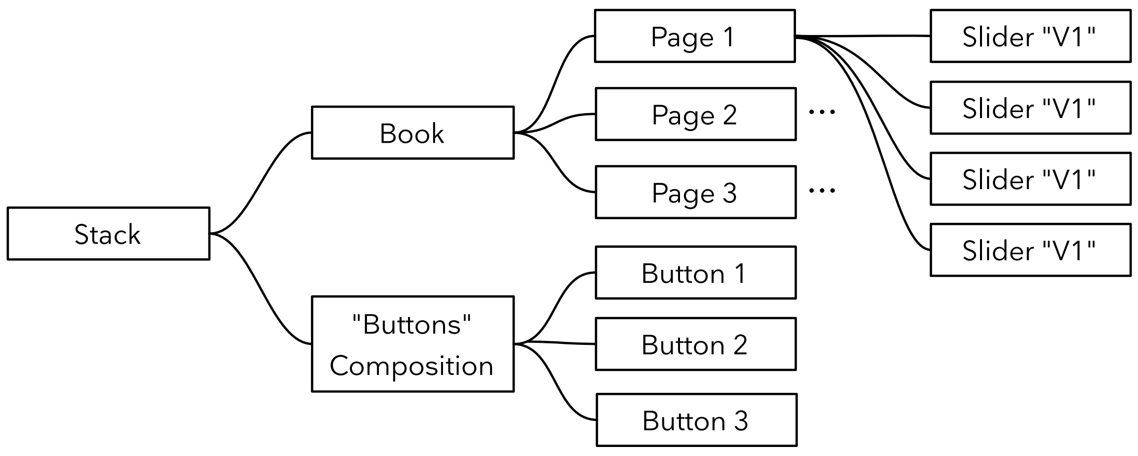
Stick to Edge

Custom HTML Classes

A Stack block with two children. The Book child (showing the four sliders) takes up most of the space, and is set to "Fill Remainder". The Composition named "Buttons" sticks to bottom edge.

- To make the stack block fill the full height of the device used to show it, leave the Stack block's Height field blank. The height will then be adapt to the device being used.

A child block can be set to **Stick to Edge**, as is the case with the "Buttons" in the example shown above. Such a child sticks to the top edge if it's placed at the top of the list, or to the bottom edge if placed at the bottom. The "Book" block, displaying the four sliders in the example above, is then set to **Fill Remainder**.



- While you can create a similar arrangement using a Composition instead of a Stack, the Stack guarantees that a top/bottom block (set to "Stick to Edge") is always positioned at the top/bottom of the screen, regardless of the screen size, with the main area taking up the remaining space. This makes a Stack more adaptive to different screen sizes on mobile devices, while all children of a Composition have fixed positions.

Scroller

A Scroller block lays out its children horizontally or vertically, allowing the viewer to manually scroll the resulting layout in the direction specified. It can also be used with a single wide or tall block, allowing it to be scrolled. Child block may appear seamlessly next to each other, or with a specified gap in between.

◆ Since this block is explicitly made for manual interaction, videos placed on it will not play automatically. Furthermore, if you display multiple videos, starting one automatically pauses any other already playing video.

- **Scroll Direction** can be set to Horizontal or Vertical, with vertical scrolling arranging the child blocks as a list of rows.
- **Gap** adds a gap of the specified number of pixels between each child.
- **Position Snap** attempts to position a child block at specified position (Center or Top/Left) when scrolling ends. Default is *Off*, for no scroll snapping.

Sizing and Scaling of Scroller and Child Blocks

You must specify the dimension perpendicular to the scroll direction. That is, you must set an explicit height of a horizontal Scroller. Child blocks of the Scroller should generally have a dimension matching this specified height or width. The Scroller itself will be scaled to fit within its enclosing block. In some cases, such as when used as the main area of a Stack block, this sizing is dynamic, using whatever space available, which will vary with the resolution and orientation of the display device.

You don't have to specify the size of child blocks corresponding to the scroll direction. That is, if the Scroll Direction is Vertical, you can leave the Height unspecified. In this case, the height will adapt to the contents of the child block. This is useful if the content is of variable size, such as dynamic text or a URL Image. Specify a value to enforce a particular size.

Child Replication

When using child replication in a Scroller, the replicants are arranged as a scrollable list. Use a single Composition as the prototypical child, with desired items – including any items bound to properties of the data source – placed inside that Composition.

Grid

Similar to the Scroller in that children are laid out as a horizontally or vertically scrollable list. However, the Grid arranges its child blocks first as a column (for a horizontal grid) or row (for a vertical grid), fitting as many children as possible before continuing in the scroll direction. To render a uniform grid, all child blocks must have the same size.

Child Replication

The Grid block is particularly useful in conjunction with child replication, where you don't have all the child blocks up front, and the number of children is dictated by the data source. In this case, you add a single Composition child, with a fixed size. Inside this child, add dynamic text, images (using Media URL) and other controls bound to the replicated data source.

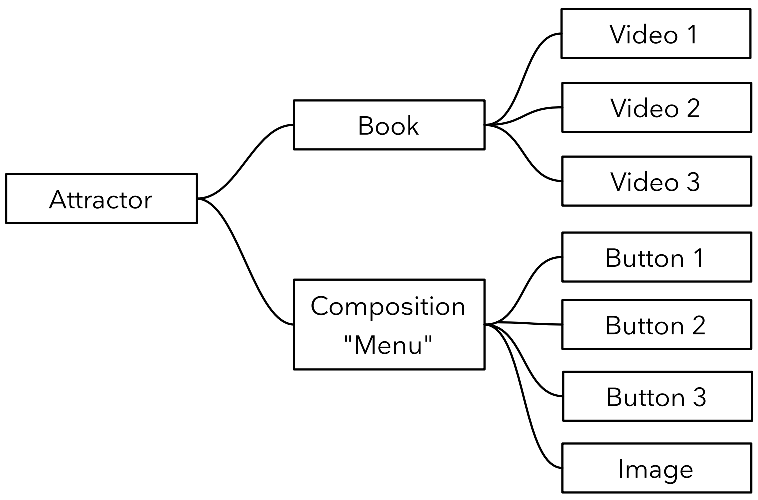
A Grid block showing images originating from a collection database.

Attractor

An Attractor has one active child, which is always its first child, optionally followed by one or more passive children. It is useful in interactive applications, where you often want to have something on screen to attract visitors. Once the visitor begins to interact, the Display Spot switches to its active state. When the active state ends (for example, when a video stops playing), the Attractor reverts back to its passive state.

- **Auto-activate with Spot** switches the Attractor to its active state when the Spot's *active* property is set, which can be done either manually (from a button or task) or automatically by a Locator navigating to this Spot.
- **Activate by Touch/Click** switches the Attractor to its active state by touching or clicking it.
- **Touch/Click Keeps Active** maintains the active state for as long as interaction recurs.
- **Auto-play on Active** starts content playback (e.g., Video, Audio or a Slideshow) automatically when made active. Uncheck if you prefer to then start content playback manually for each such child block.
- **Deactivate on Play End** terminates the active state once playing content stops, possibly after some delay.
- **Linger after Play** adds additional time before deactivating due to *Deactivate on Play End. This is u*seful if you have several playable items visible in the active state and need some time to start playing another item without the Attractor reverting to its passive state while you make up your mind.

- **Deactivate on Timeout** makes the Attractor revert to its passive state after a specified amount of time. This is useful if there's no playable content controlling the duration.
- **Transition** specifies a transition used to switch between active and passive states.
- **Transition Delay** allows the start of the transition to be delayed by the specified amount, giving more time to the upcoming content to load.
- **Reverse Transition on Active** inverts the direction of directional transitions (such as Slide or Push) when transitioning from passive to active state.



An Attractor block with a Book as its active child and a "Menu" Composition as its passive child.

The example above shows an Attractor combined with some other blocks to create an interactive kiosk. Here, the *passive* state displays a menu. When a visitor interacts by pressing a button on the menu to start a video, the Attractor automatically switches to its *active* state, showing one of the videos in the Book. The Book is the first, hence the *active*, child of the Attractor. Once the video is finished, the Attractor automatically reverts to the passive state, showing the "Menu" composition.

The spot can be made active by one of the following events:

- By someone touching it. Applies only when "Activate by Touch/Click" is selected.
- By using a Locator to access the Spot. This allows a visitor to trigger the active state using a mobile device
- Programmatically, by setting the spot's *active* property to *true*.
- A *Reveal* behavior showing a block under the active child.
- By navigating to such a child block by any other means, such as a button or external command.

The spot, and its Attractor, reverts to its passive state by one of the following:

- When reaching the end of playback (video, slideshow, etc). Applies only when "Deactivate on Play End" is selected. Here you can also specify a "Linger after Play" time, to provide some time to pause, seek or play some other content before reverting to the passive state.
- Programmatically, by setting the spot's *active* property to false, a Reveal behavior revealing a block in a passive child or by navigating to such a child block by any other means, such as a button or external command.
- After a certain time. Applies only when "Deactivate on Timeout" is selected, once the spot has been idle for the specified amount of time. Any interaction, such as touch gestures, can reset the timeout, keeping the Attractor in its active state.

◆ **NOTE:** Interaction inside a Web Block can not be detected, and will not automatically be recognized as activity. However, if you have control over the page displayed by the Web Block, code can be added there to deliberately notify Blocks about any interactions, thus keeping the Attractor in its active state.

Using multiple or no passive child blocks

While you often have just a single passive child, as in the example above, you may have any number. If there are multiple passive child blocks, the Attractor will alternate among them, which may offer some variation to the attract loop, rather than always showing the same passive block.

Alternatively, you may have no passive block at all. In this case, the Attractor will still affect the active state of the Spot, but no content change will occur when switching between its active/passive state. This is occasionally useful to trigger some other action (e.g., a Task) after some time of inactivity.

Using Nested Attractors

You may occasionally want to use more than one Attractor at the same time. An inner Attractor can, for instance, be used to manage a menu, where buttons are used to play a number of videos. Once such a video stops, the Attractor is used to return to the menu page. This nested Attractor/menu is then part of a larger structure, possibly housing multiple such menus. Outside all these, theres another Attractor that manages the active/passive mode of the entire spot, returning it to some neutral state after a timeout. In this scenario, only the outermost Attractor will be governed by and control the spot's *active* property.

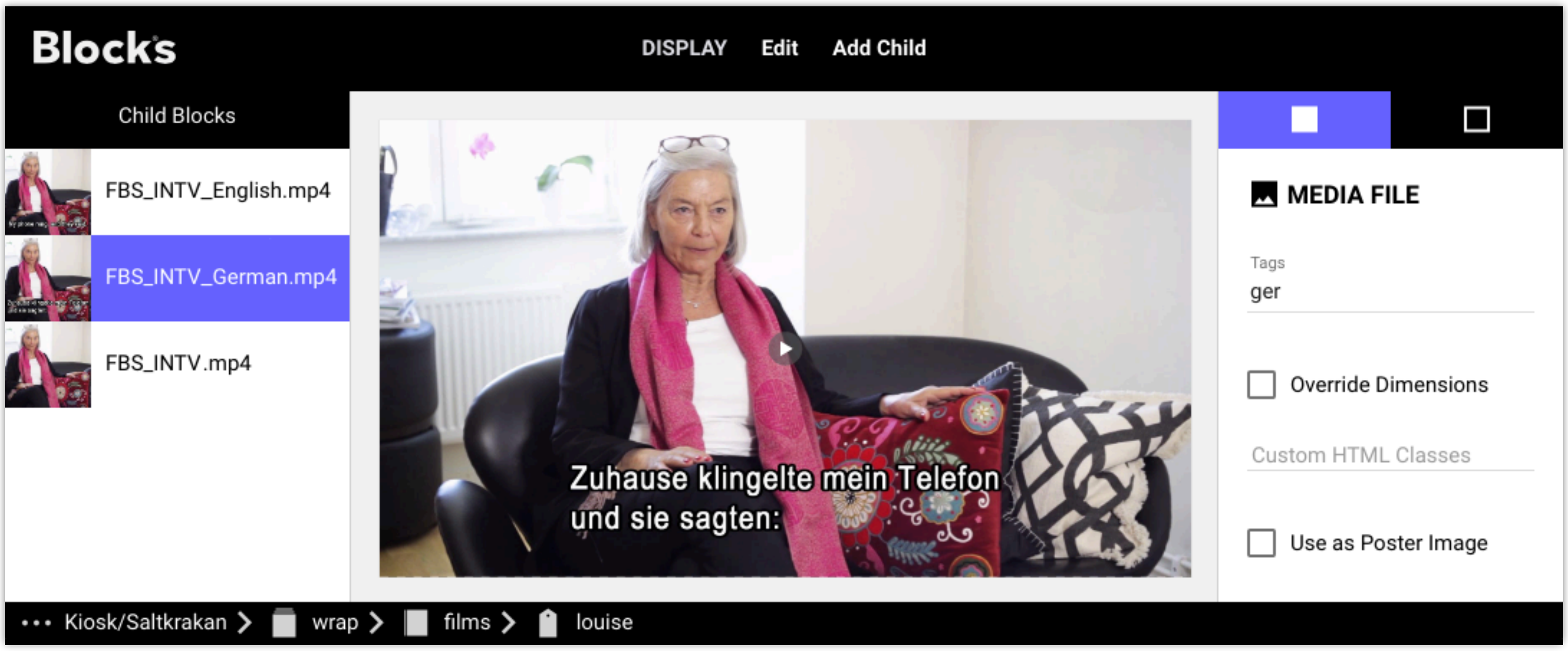
◆ The active state of the outermost Attractor corresponds to the *active* property of the Display Spot presenting the block. This binding is bidirectional, so that you can force the attractor to its active or passive state by setting the spot's *active* property. Note that this applies only to the outermost Attractor if you use multiple nested Attractor blocks.

Tag Selector

The Tag Selector block works in conjunction with tags assigned to Spots. Tags can be assigned statically (as described under "Tags" in the chapter on Spots), or they can be assigned dynamically and interactively, using buttons and tasks. Typical uses of tags include:

- Indicating that a Spot is likely to be a mobile device by statically assigning it the "mobile" tag.
- Indicating the relative position of Spots inside a Spot Group; e.g. "left", "center" and "right", thereby making the correct content show up on the appropriate display in the group.
- Indicating the age group or language preference of visitors. A menu page can be shown on the mobile device, such as the one shown to the right, allowing visitors to select the preferred language from a set of supported languages.

Regardless of how a tag is applied to a Spot, it can then be used by a Tag Selector block to select the best matching child to show from its set of child blocks.



A Tag Selector block shows the most appropriate child. Here, a language tag is assigned based on the visitor's language preference, subsequently selected by the "ger" tag for playing the German version of the video.

Using Multiple Tags

You can apply any number of tags to a Spot. Likewise, you can specify any number of tags for Tag Selector child blocks. When displaying the Tag Selector, it will show the child with most matching tags. If no tag matches, it will show the first child block.

Overriding Dimensions of Child Blocks

Tag selector (as well as Synchronizer) child blocks have a setting to "Override Dimensions". This is useful when you have a single block that should play on devices with different size screens, where the tag is used to select the proper screen type. This can be used to differentiate between landscape or portrait orientation, or between displays and mobile devices. In those cases, select "Override Dimensions" for the child block that should be used with other dimensions than the root block, then enter the desired dimensions.

Media File

This block type represents a single media file. Add it either by choosing "Media File…" on the Add Child menu, or simply by dragging the media file into the child block list.

You can use the following media file types:

- JPEG, PNG, SVG and GIF images.
- Video encoded using the H.264 codec, typically in an MP4 file (see below for more on video file encoding and formats).
- Audio encoded using the AAC codec, typically in an m4a file, or similar.
- A 3D model in the GLB file format. This is only allowed inside a 3D block.

In general, images and video should have the same dimensions as the block to which you add them. For instance, if the block is 1920x1080, images and video should have the same size. The exception is the Composition block, where images and other child blocks are often smaller than the composition itself, in order to be placed side by side or overlapping. While a Composition allows you to scale child blocks, you're usually better off producing the content at the correct size to begin with, as doing so will result i better quality and less strain on the playback device. Particularly for video or large images.

◆ More on video encoding and the use of other file formats here.

Replacing Media Files

Replace the file used by a child Media File block by selecting the block and choosing "Replace Media" on the Edit menu. For a top level Media File block, replace by dragging another media file of the same type and size to the block on the Display page. Replacing the file in this way retains other settings.

Playback

Audio and video media have the following additional settings:

- **Play Automatically** starts playback as soon as the media block appears. Otherwise, media playback must be started manually by clicking a play button that appears at the center of the media block.

  ◆ With some web browser, automatic media playback may be restricted. This does not apply to PIXILAB Player. This article has more information on how to make sure that content can play without any user interaction. Some mobile devices (e.g., iOS) does not allow playback of media with audio without user interaction.

- **Loop** causes media to play repeatedly. Otherwise, media plays only once.

- **Pause without Stopping** lets you pause video playback without causing any enclosing Attractor, Locator or similar to close, allowing you to resume playback later.
- **Auto-stop after** (only available when "Pause without Stopping" is selected) ends any paused state after the specified time, thus avoiding leaving the spot in a paused state indefinitely.
- **Exclude from Sync** is occasionally useful for video or audio that for structural reasons must be contained in a Synchronizer context but that should *not* partake in synchronization. This can, for example, be a looping background sound clip or an animated logo.
- **Playback Controls** controls whether a progress bar will appear during playback. *None* hides the progress bar and play button, *Auto* shows the play button it while paused, *Progress* shows Blocks' standard progress bar (same look regardless of browser/player type and can be styled with CSS) or *Native* shows the browser's native progress bar.
- **Show Poster Image** shows a still image placeholder until playback is started. Uncheck this option to show nothing at all, thus allowing any content positioned behind the video to appear instead.

Local Sync Slave

This option is available only when inside a Synchronizer. If selected, the media will synchronize only *within* the current Spot. This is useful, for example, to synchronize two videos (or a video and a Subtitle block) playing on the same Spot. If so, place those inside a Synchronizer, then select this option for the one to be slaved.

◆ If no external synchronization is to be used at all, then also select "Local Synchronization" for the enclosing Synchronizer.

Video Playback Settings for iPhone

Video files have an additional setting, controlling their behavior iPhones. It has no effect on other display devices.

- **iPhone Video Fullscreen** automatically zooms the video to full screen size while playing.
- **iPhone Video Inline** causes the video to remain in its current position on screen, and play there. This allows other adjacent elements to remain visible, but also result in the video remaining small on screen, unless explicitly zoomed to full screen size by the user (by tapping the zoom button located in the corner of the video).

Media URL

While similar to the Media File block described above, this block obtains the media to be shown based on a URL rather than a pre-existing file. This allows the media shown to vary, as dictated by the value of a bound property. This has two primary use cases:

- To control what's shown at runtime. Useful when the media to show isn't available at block creation time. This is done by binding the URL to any system property (including spot parameter) that can provide a valid URL.
- To display media referenced by a URL in a replicated block, where the URL originates from a content management database providing the data.

While most settings are identical to those described above for the Media File block, this block has a few additional settings:

- **Media Type** indicates what type of media (still image, video, audio) you expect to receive.
- **Aspect Ratio Adaptation** determines what to do when the aspect ratio of the media shown doesn't match the size of the block. *Stretch* scales the media non-uniformly. *Fit Inside* and *Crop* maintains its aspect ratio, either by letterboxing or cropping the media.
- **Image URL Binding** specifies the URL of a still image, or – in the case of video – any poster frame to show prior to starting video playback.
- **A/V URL Binding** specifies the URL of any audio or video media.
    - ◆ **NOTE:** If you don't know beforehand whether image or video will be available, set *Media Type* to *Video*. Then provide an *A/V URL* if you indeed get video (possibly along with a poster frame image as the *Image URL*). Alternatively, if all you get is a still image, just provide the *Image URL*, leaving the *A/V URL* blank, thus rendering the image only, without any video controls.

Camera

The Camera block allows you, or a visitor to take a photo. This uses either a webcam connected to a Display Spot or the mobile camera functionality of a Visitor Spot.

> ◆ **IMPORTANT:** This is a premium block type, requiring additional license options. For Displays Spots *not* using our PIXILAB Player software, a secure server connection (https) is required to access the camera. This requirement does not apply to Visitor Spots, where the device's native camera functionality is used instead. For use on Visitor Spots, you must also have the Visitor Data Collection license option.

The resulting image is then uploaded to the Blocks server, where a User Script is needed to receive the image by subscribing to the 'image' event. The User Script will then have to decide what to do with the image. It could, for example, be associated with an individual visitor as part of Visitor Data Collection.

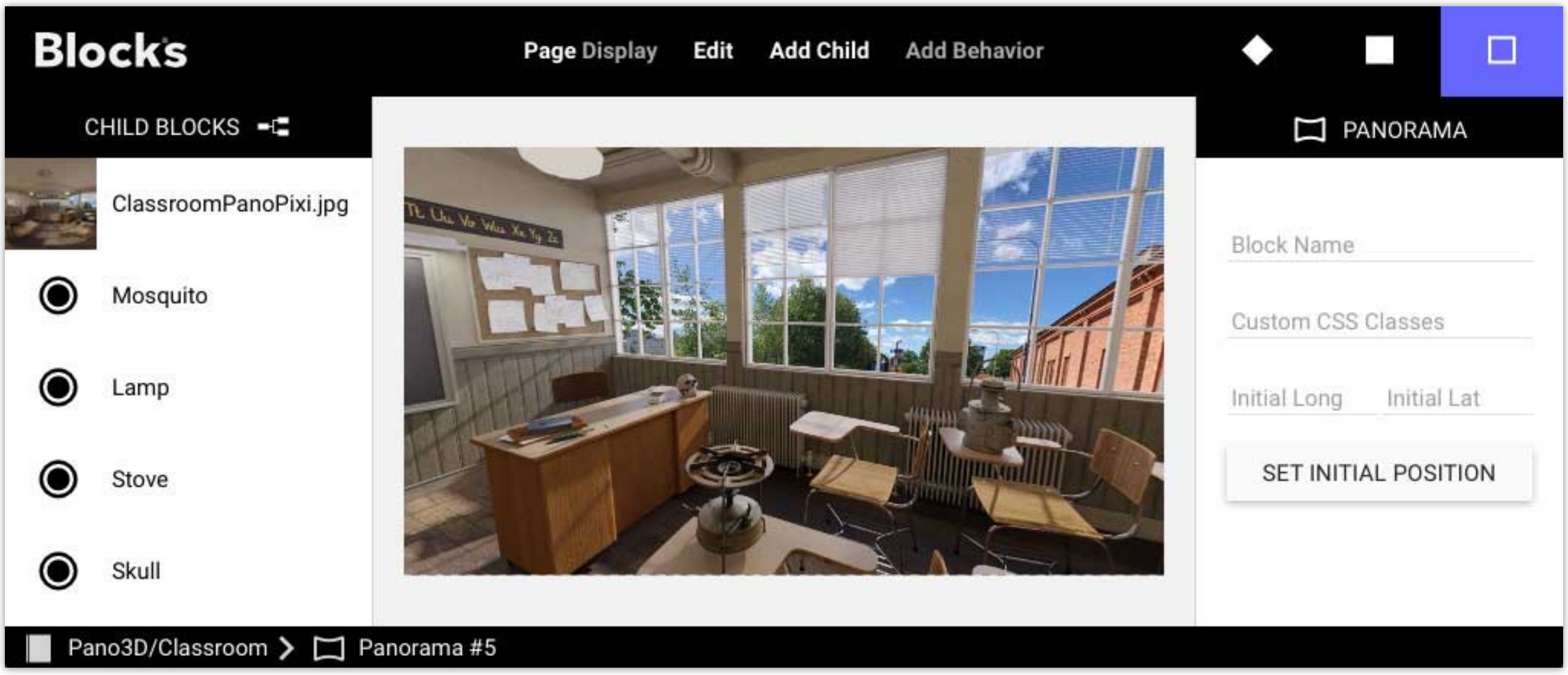
- **Start Automatically** starts the camera in viewfinder mode immediately, without any user interaction. This option applies only to Display Spots running PIXILAB Player, and can not be used on Visitor Spot (i.e., mobile devices). If not selected, you must press the camera symbol once to activate viewfinder mode before an image can then be captured by a second press.
- **Mirror Camera View** flips the camera's viewfinder horizontally, often desirable for a visitor-facing camera. This setting does not apply on Visitor Spots, where the native camera typically has its own corresponding setting.
- **Photo Source on Mobile** decides whether a visitor must take a new photo using the camera (Camera Only) or can either take a new photo *or* chose an existing one from the device's photo library (Camera or Library).
- **Camera Rotation** allows a webcam to be mounted sideways. Many webcams are designed to capture a wide aspect ratio image. If you prefer a tall image, mount the camera rotated 90° and change this setting accordingly to counter-rotate the preview and resulting image.
- **Camera Roll Name** provides an additional name to the User Script used to receive the image. This is useful if you need to show a Camera block in more than one context, as a means of indicating the context to the script.

> ◆ **HINT:** By combining "Start Automatically" with a Press behavior, you can control when images are captured programmatically, rather than having someone pressing the on-screen camera symbol.

Panorama

The Panorama block presents a 360 degree environment, such as a room or an outdoor scene – useful if you want to bring your audience to a place they can't visit in person. It's based on a 360 degree spherical image or video, sometimes referred to as an equirectangular image. This provides a panoramic view from the vantage point of the camera used to take it. Look around by swiping, pinch to zoom in/out. Buttons and other child blocks can be added to the inside of this sphere. Buttons can act as click-able hot-spots, navigating to other locations.

◆ **NOTE:** This is a premium block type, requiring additional license options in order to be displayed.



A Panorama block with its equirectangular background image and a few buttons.

To specify the initial orientation of the panorama when first shown on a spot, spin it to the desired view, then press "Set Initial Position". Select "Reset after Timeout" to reset the panorama to its initial orientation after a configurable period of inactivity. When selected, you can also opt to have the panorama auto-rotate while inactive by specifying a rotation speed.

Note that you can only add a single, equirectangular background image or video, which must be the first block in the list of child blocks. This must have a 2-to-1 aspect ratio. Find some example videos here. An equirectangular image (or sometimes video) can be made in one of the following ways:

- Use specialized a camera with a 360 degree lens.
- Use the Google Street View Android/iOS app to make the image, then transfer it to your computer.
- Render one from a 3D program (if your entire scene is a 3D model).
- Make/edit one in Adobe Photoshop by stitching several still images together.

Adding Child Blocks

You can add child blocks and controls to a Panorama. For example, you can have a small video playing in a window appearing in the panorama. Buttons added to the panorama act as hot spots, triggering an action when clicked. Named child blocks can be used to orient the panorama facing in that direction using a button, a Reveal or Navigate behavior or the *gotoBlock* function.

Position the child by dragging it to the desired location, specified as the latitude and longitude inside the sphere. The dimensions must be set numerically. Use custom CSS to change the look of buttons and other controls, if desired. Program the button as usual, making it go to another view using a Book, controlling some external function, or anything else a button can do in Blocks.

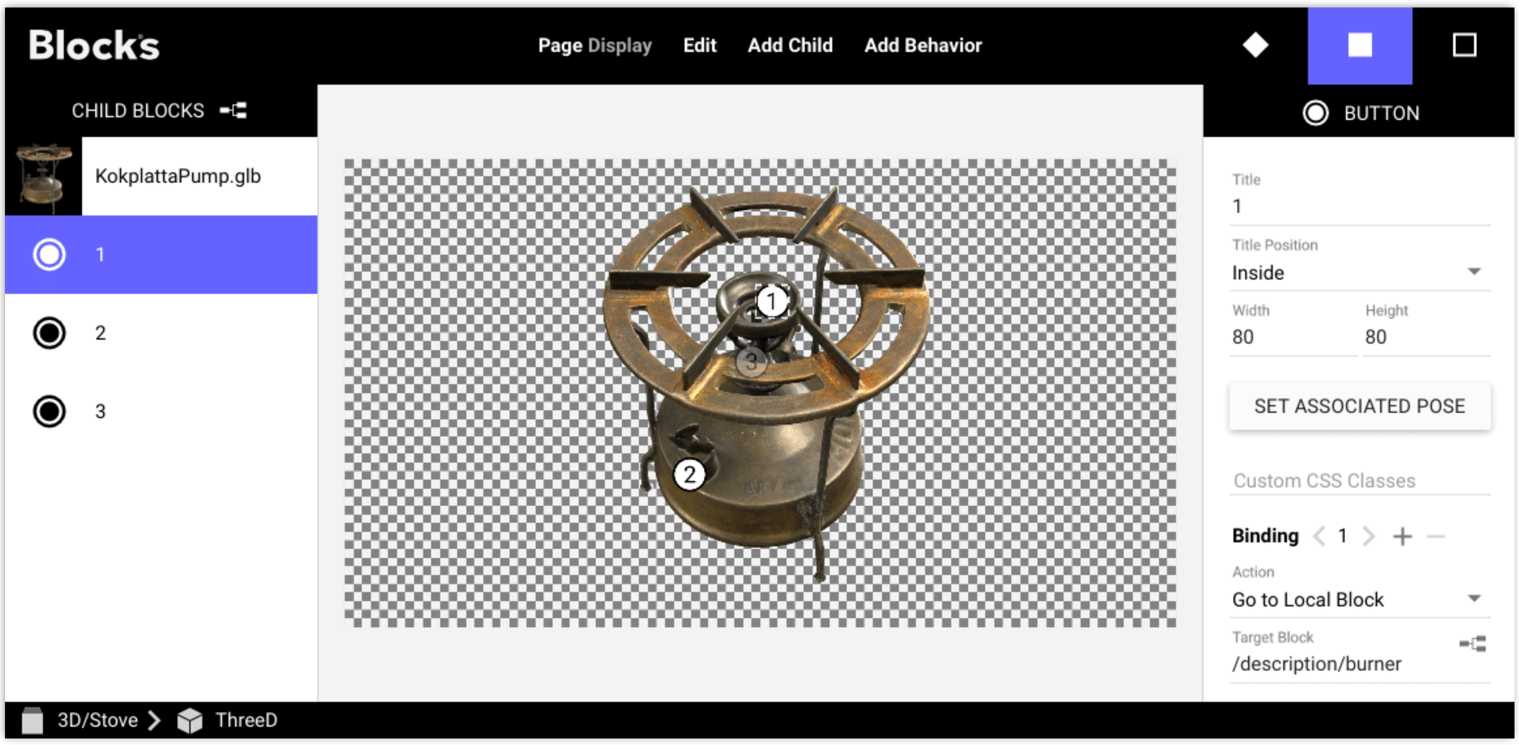
Distort with Perspective
Select this option to make the shape of the button follow the perspective of the panorama – particularly useful when using a larger, rectangular and invisible button to make an area in the image clickable, since the shape of an area is distorted when close to the edges of the viewport.

Parallel to Ground
Flips the child block 90 degrees along its horizontal axis, making it appear as if applied to the floor.

3D

This block type [displays a 3D model]() that can be rotated, moved sideways and zoomed interactively. Use it to show a 3D scanned artifact, or a model originating from some 3D or CAD application. It's based on a 3D model in the [GLB format](), which is a standardized 3D format that encapsulates the geometry, textures, animation and other related data into a single, compact file. Buttons can be added to the surface of the 3D object, acting as clickable hot-spots.

- ◆ **NOTE:** This is a premium block type, requiring additional license options in order to be displayed.



Title: 1
Title Position: Inside
Width: 80
Height: 80
Action: Go to Local Block
Target Block: /description/burner

A 3D block with its model and a few buttons.

Visitors can navigate the 3D block using touch (recommended) or mouse.

Navigation	Touch	Mouse
Spin around	Single finger drag	Draw using left mouse button
Zoom in/out	Pinch using two fingers	Scroll wheel
Move sideways or up/down	Drag using two fingers	Press scroll wheel and drag

Outer Block Settings
- **Illumination Factor** controls the intensity of the standard [three-point lighting]() automatically applied by Blocks. Adjust this setting if your model appears over/under exposed. Set this factor to zero or a low value if your model includes its own lighting inside the 3D file.
- **Show Shadows** makes the key light cast shadows on the model. This may slow down rendering of complex models.

- **Tilt Min/Max** controls limits the ability to rotate the model around its horizontal axis. This is particularly useful for models where only one side (typically the top) has been scanned. Then set Tilt Min to a small negative value, such as -10 degrees.
- **Set as Initial Pose.** Specify the orientation and size of the model when appearing on a spot by positioning it in the editor, then press “Set as Initial Pose”.
- **Reset after Timeout** automatically resets the model to its initial pose after a configurable period of inactivity. When selected, you can also opt to have the model auto-rotate while inactive by specifying a rotation speed.

Obtaining 3D models

Note that you can only add a single 3D model, which must be the first block in the list of child blocks. This model must be a GLB file. A model can be obtained in a number of ways – often requiring a combination of multiple techniques.

- For physical objects, where a high degree of precision is desired, use a professional grade 3D scanner, available from a number of manufacturers (such as this one). These often combine multiple techniques, such as photography and lidar
- Photogrammetry, deriving the 3D model from a large number of photographs.
- For quick tests, you may be able to use a phone-based scanner app.
- Create or edit a model in a 3D application, then export it as a GLB file.
- Download one from one of the many sites on the internet that provide 3D files, such as Sketchfab or the Smithsonian.

Sketchfab is a great website for finding 3D models. Unfortunately, GLB isn’t one of their supported formats. It does, however, provide the closely related GLTF format. This can easily be converted to GLB using a number of online services, as well as many 3D programs, such as Blender. The Smithsonian museum has lots of 3D models in the GLB format.

◆ **NOTE:** 3D files can very large, and may need to be optimized for best performance. This applies in particular if you want to use 3D-scans in Blocks. Blocks supports the DRACO compression method, often useful for complex 3D models.

Adding Buttons to a 3D Block

Add buttons to the 3D model using the command on the Add Child menu. Position the button by dragging it to the desired location. The dimensions of a button must be set numerically. Use custom CSS to change the look of buttons, adding icons or other symbols, etc. Program the button, making it go to another view using a Book, controlling some external function, or anything else a button can do in Blocks.

◆ **HINT:** Buttons in a 3D block can only be positioned on the surface of the model. To place buttons outside the model, wrap the 3D block with a Composition, then add buttons there instead.

A button can also be used to position the 3D object it's attached to:

- Select a button in the 3D block.
- Position the 3D model as desired.
- Press “Set Associated Pose”.

Now locate that button using, for example, another button set to “Go to Local Block” targeting the button that has the associated pose, or a Task telling the Spot that shows the 3D model to locate the path to the button.

◆ **HINT:** If you only want to use a button for positioning, you can hide it or change its appearance using CSS.

Magazine 🕮

The Magazine block type displays its child blocks as pages across a two-page spread, like a magazine or book. You can turn pages manually, by dragging/tapping a page corner, or programmatically using buttons, tasks, etc.

- ◆ **NOTE:** This is a premium block type, requiring additional license options in order to be displayed.

The Magazine block has the following options:

Flipping Duration specifies how long each automatic page turn animation takes, in mS.

Start Page Offset lets you start out at another page than the very first one when the Magazine appears.

Show Cover renders the first child block as a right hand side cover page, rather than as the left hand side of the first spread.

Rigid Pages renders the page flip animation as a "cardboard" page, rather than a soft, curly page.

Flip by Clicking lets you flip pages by simply clicking on the next/previous page in addition to dragging its corner.

Flip Hint Animation briefly reveals the corner of the next page to indicate it can be flipped.

Hint Animation Interval controls the time between flip hint animations, in seconds.

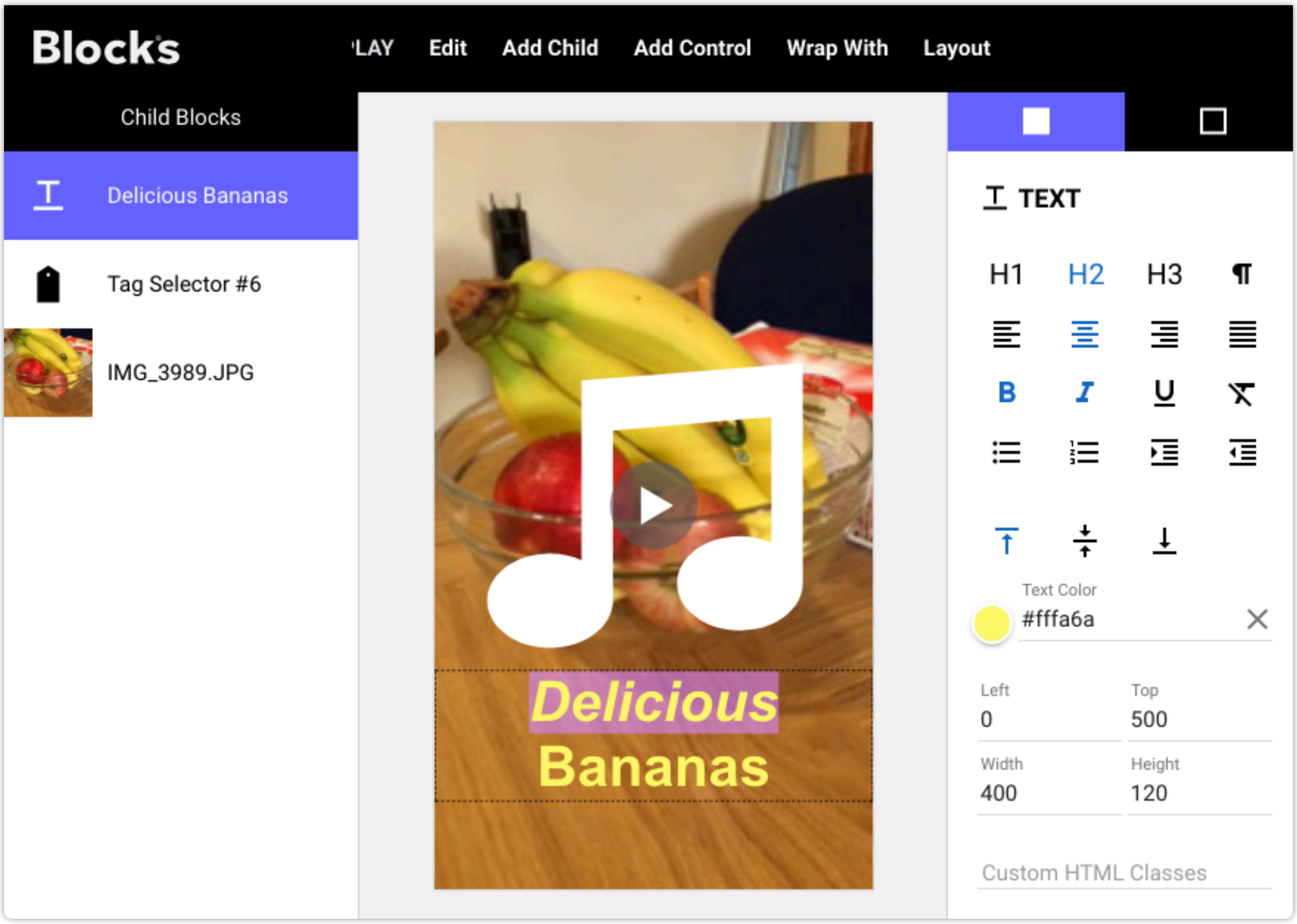
Delay after Interaction specifies how long it will take after a page flip until the hint animation starts.

Show Page Numbers renders a page number at the bottom corner of each page, with two styling options.

- ◆ **NOTE:** Keep in mind that each spread consists of *two* child blocks. So to use buttons or other programmatic means to flip pages, make those go forward or backwards by *two* pages by putting /+2 at the end of the target block path.

Text

Use a text block to add arbitrary text to your presentation.



Text Color: #fffa6a

Left: 0

Top: 500

Width: 400

Height: 120

A text block with the text "Delicious" selected.

Position and size the Text block inside the Composition using the mouse, or numerically. Double-click the Text block to edit its text. Use the formatting buttons on the right hand side to change size and position, as well as other attributes of the text. The H1, H2, H3 buttons change the text size to three predefined heading sizes. The "paragraph" button reverts the size to normal text size. The second row of buttons control the horizontal alignment of the selected text. The third row applies variations, such as **bold** or *italics* (with the last button removing any such format overrides).

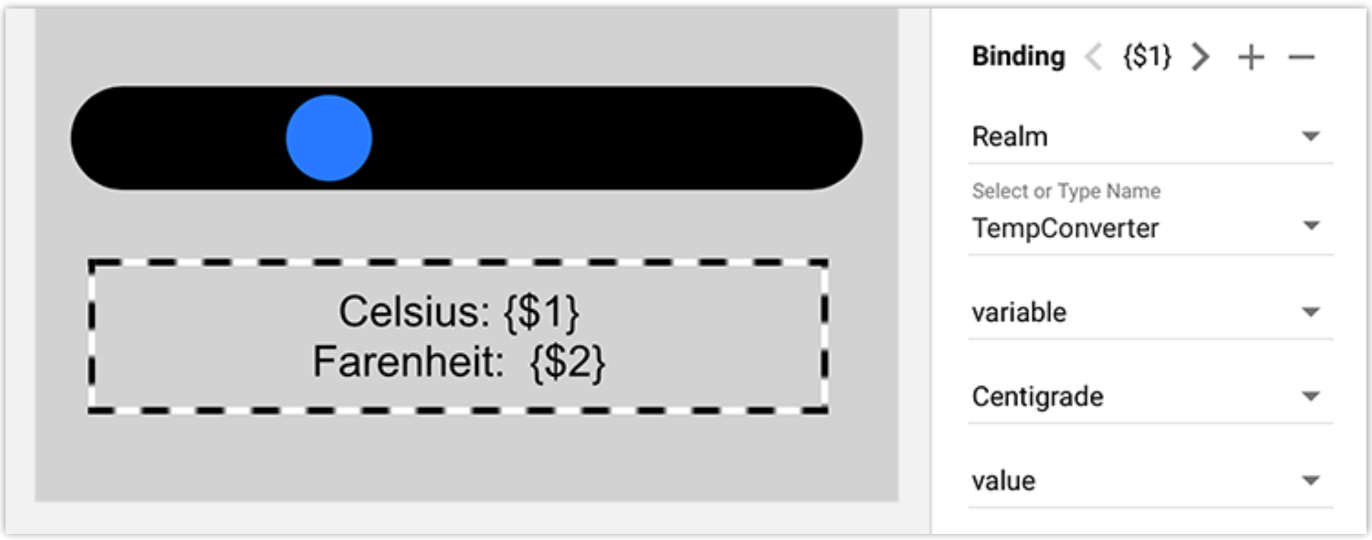
- Other fonts, sizes or variations can be added by means of custom CSS.

The fourth row of buttons can be used to add bullets or numbers to paragraphs – often useful when displaying lists – and to apply indentation. The last row controls the vertical alignment of the text inside the Text block.

- A Text block can not be added as direct child in all places. To use a text block where Text doesn't appear on the Add Child menu, first add a Composition then add the Text block inside that composition.

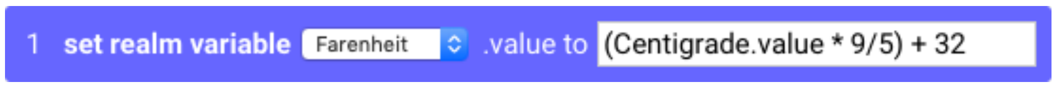
Dynamic Text

In addition to static text, as described above, the Text block can also display dynamic text originating from bound properties. This is similar to how Indicators and Bars work, but shows the value as text rather than graphically. After adding a property to a Text block, the property can be referenced inside the text using a special syntax, consisting of a pair of curly braces containing a dollar sign and a 1-based property index number. The Text shown below includes two such dynamic properties.

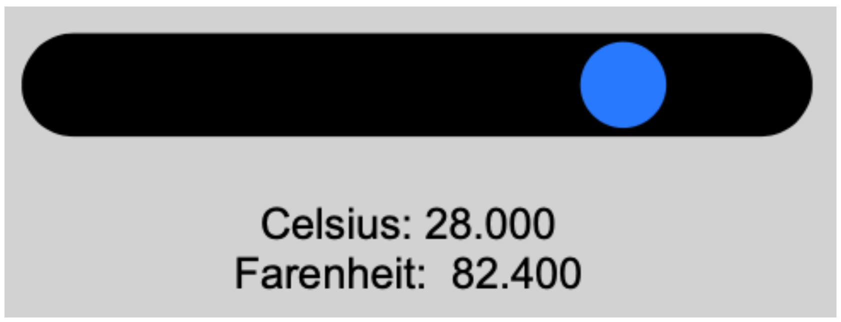

Celsius: {$1}
Farenheit: {$2}

Binding {$1}
Realm
Select or Type Name: TempConverter
variable
Centigrade
value

The first property is bound to the value of a realm variable named Centigrade, inserted into the text as {$1}. Likewise, the second one is bound to another realm variable named Fahrenheit, and is inserted as {$2}. The slider above the text is bound to the same Centigrade realm variable. A task is triggered when the Centigrade variable changes, setting the Fahrenheit variable based on the Centigrade variable, like this:


```
1  set realm variable Farenheit .value to (Centigrade.value * 9/5) + 32
```

Displaying the block containing the above on device with a touch screen, you can change the values using the slider, showing the temperature in both Centigrade and Fahrenheit.


Celsius: 28.000
Farenheit: 82.400

Formatting Dynamic Text

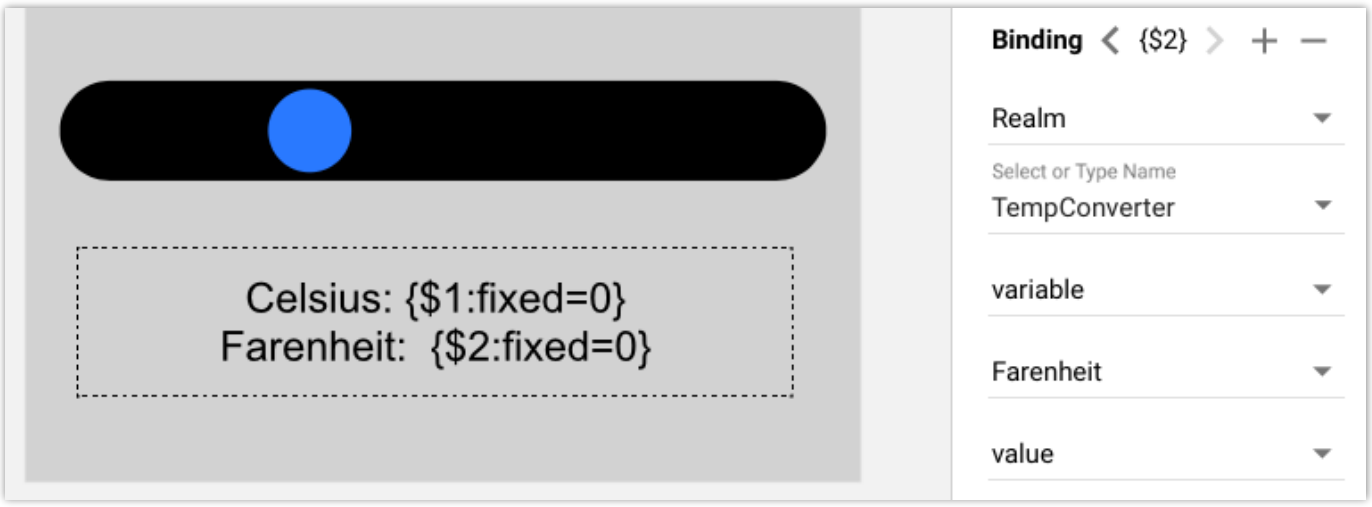
In addition to the property index number, some data types also accept additional formatting options. These options are separated from the property index (as well as from each other if using more than one) by a colon. Each formatter consists of a name, and equal sign and a value. Applicable formatters depend on the type of value being presented.

Name	Applies to	Value
true	boolean	Text to show for *true* value.
false	boolean	Text to show for *false* value.
scale	number	Factor to multiply by.
fixed	number, time	Number of fraction digits to show.
time	time	Combination of *hmsf* characters, indicating which time components to include.
freq	time	Frequency, in hertz, at which time value should be updated.
locale	number	Applies locale-specific formatting based on the parameter, e.g. en-GB for english style.

A boolean value is by default rendered as *true* or *false*. If you instead want it to be shown as *YES* or *NO*, reference the property like this:

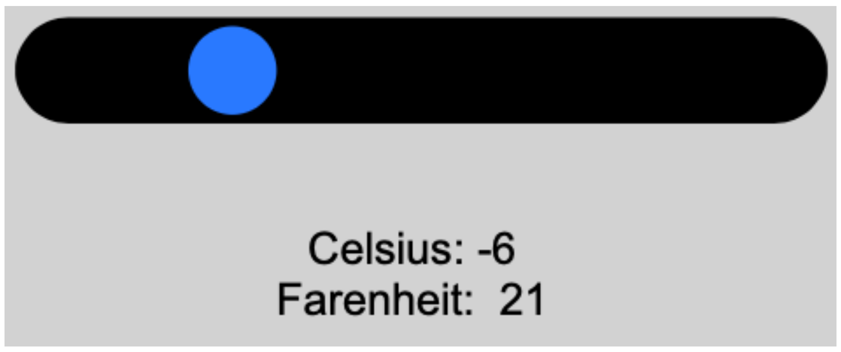
```
{$1:true=YES:false=NO}
```

A number can be formatted using a *scale* factor (e.g., to show a normalized 0…1 value as a percentage 0…100) or using a *fixed* number of decimals. You can use this in the example shown above to suppress the decimal digits following the temperatures like this:



Celsius: {$1:fixed=0}
Farenheit: {$2:fixed=0}

Here, the number of fractional digits is set to zero, resulting in less confusing values:



A time value, such as the time position of a video or a WATCHOUT timeline, can be formatted using the *time* formatter followed by the desired combination of *hmsf* representing hours, minutes, seconds, fractions. For example, the following formatting specification will show minutes and seconds:

```
{$1:time=ms}
```

Locator

The Locator block, unlike most other block types, doesn't show any content of its own. Nor can it have any child blocks. When displayed, all you'll see is an (optional) numeric keypad. The purpose of this block is to determine *where a visitor is* in order to present content associated with that spot.

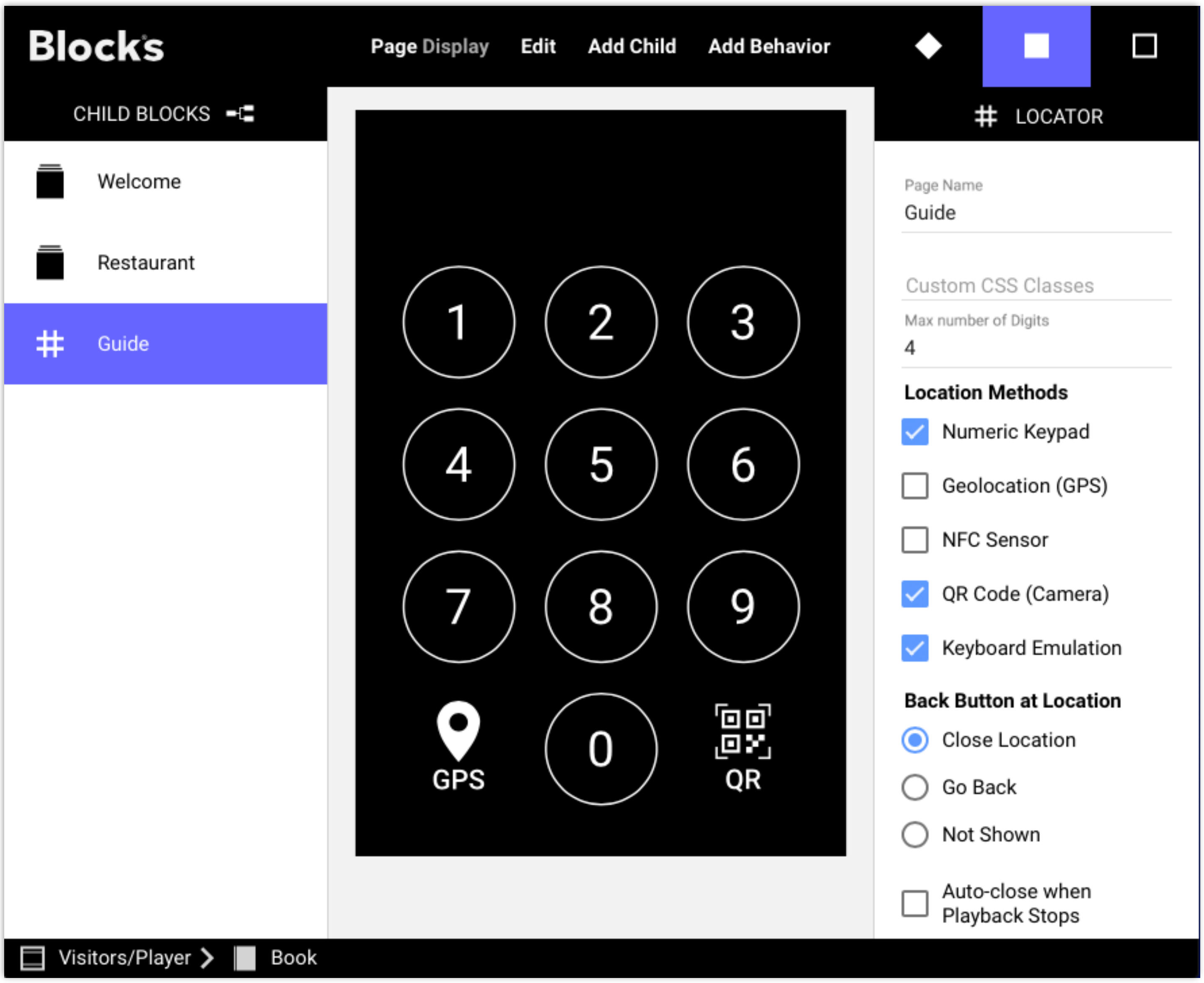
◆ **NOTE:** This is a premium block type, requiring an additional license option in order to be used.

The visitor's location can be obtained by any of the following means:

- By entering a number using the numeric keypad. This number correlates to a Location ID that can be assigned to spots.
- By GPS position. This position is matched agains geolocation zones specified in spots, and can be used only outdoors with devices that have a built-in GPS receiver, such as most mobile phones.
- By scanning a QR code found at the spot. This can be done using the camera built into most mobile devices. Alternatively, it can use specialized RFID/QR scanning hardware emulating a keyboard (such as the Zebra range of Android-based scanners).
- Manually, using an on-screen button to specify your position. The button can show a picture of the object, allowing you to select it from a menu of objects.

◆ **NOTE:** The GPS and camera-based QR code options can only be used when accessing Blocks over an https connection. This is an advanced server configuration option, requiring a valid https certificate.

Depending on the type of Spot being located, the content can be anything from a text or an image, an interactive multi-page presentation, a video or a sound. Often, you'll use just a sound clip, possibly accompanied by an image confirming you're at the right place. This can be used to create an "audio guide," playing audio localized to where you are, or describing a particular object.



Page Name
Guide

Custom CSS Classes

Max number of Digits
4

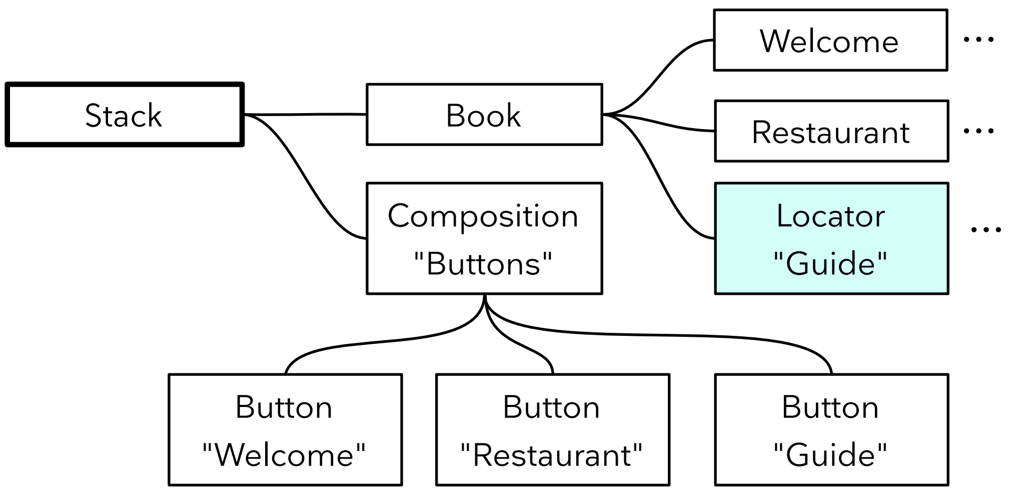
Location Methods
- [✓] Numeric Keypad
- [ ] Geolocation (GPS)
- [ ] NFC Sensor
- [✓] QR Code (Camera)
- [✓] Keyboard Emulation

Back Button at Location
- (●) Close Location
- ( ) Go Back
- ( ) Not Shown

- [ ] Auto-close when Playback Stops

A Locator block, with numeric keypad, GPS and QR modes enabled.

A Locator can be used inside other blocks, providing further interactivity. As indicated by the breadcrumbs at the bottom of the illustration above, the Locator is inside a Book inside a Stack block. The buttons at the bottom of the Stack block provide overall navigation (as in the example shown under Stack in this chapter). Book pages can provide interaction or other content such as a composition with buttons for choosing your preferred language, or a web block showing today's specials in the local restaurant.



A Locator being used as a part of an interactive, multi-page mobile guide.

While on the Guide page of the example shown above, the Locator's keypad is shown. Upon entering a number or using GPS/QR to determine your visitor's location, the corresponding Spot is located, and its block appears on the mobile device. If this is a Location spot, you can get additional information (including images, sound and video) related to an object at that location. If it is a Display or WATCHOUT spot, you can get synchronized audio, perhaps including narration in your visitor's preferred language.

Once the content associated with the spot stops playing, the keypad reappears. A button is also overlaid on the content while playing, allowing you to stop playback at any time.

Locator Settings

The Locator has a number of options, allowing it to be tailored to special requirements.

Location Methods

Controls which methods that can be used to determine your visitor's location.

- **Numeric Keypad** shows a keypad allowing your visitor to indicate her location by entering a number shown at each relevant location (using on-screen message or a printed sign).
- **Geolocation (GPS)** shows a GPS button. When pressed, the phone's GPS is used to determine your visitor's location. The GPS position is matched against geolocation information associated with Spots.
- **NFC Sensor** shows an NFC button. Press this button to activate the NFC reader, which can then be used to read a URL containing a spot path or location ID as a query parameter named *spot*. On Android, most browsers natively support reading tags after pressing the NFC button and granting permission. **NOTE:** On iOS, the operating system intercepts NFC, forwarding to Safari. For a *spot* query parameter to work here, the block shown by the Visitor Spot must have a Locator on its initial page.
- **QR Code (Camera)** shows a QR code button. When pressed, the phone's camera can be used to scan a QR code displayed on a display spot (using a QR Code block) or on a printed sign at each relevant location.
- **Keyboard Emulation** allows a Location ID to be entered using a keyboard-emulating RFID/QR code scanner, typing the code followed by the Enter key. This also requires the RFID/QR Scanner Input to be selected in the spot's settings.

◆ **IMPORTANT:** When used on a Visitor Spot, the Geolocation, NFC and QR-Code methods require that the Blocks server is accessed using a secure URL (https). Furthermore, these actions require user interaction and confirmation. At most two of the Geolocation, NFC and QR-Code methods may be used together with the Numeric Keypad.

If the Numeric Keypad isn't enabled, no keypad will be shown. This allows anything *behind* the Locator in a Composition to be visible, using the Locator only as a means of showing the block assigned to the Spot being located. This is useful when the location is established by other means:

- Using RFID/NFC/QR scanning.
- Using a button to perform the navigation, where the button is then set to "Locate Spot".
- Programmatically, using a Task or a script.

Back Button at Location

Controls the behavior of the button used to return from showing the block at a located spot:

- **Close Location.** Immediately closes the location, returning to the keypad or menu.
- **Go Back.** Returns to the previous view, which in most cases is equivalent to closing the location. Useful if the location's block provides further sub-navigation (e.g., using buttons), in which case this setting works like the browser's back button, navigating back one page at a time, rather than immediately closing the location.
- **Not Shown.** Hides the back button entirely. Useful when end-of-play is controlled by other means, such as a task or a script, or when you want to use a regular button (with its Action set to "Go Back" or "Close Locator") to close the location.

Auto-close when Playback Stops

Controls whether the Locator will return to the numeric keypad when playback ends. Normally, that's what you want. However, if a located target contain several individually playable items, you may prefer to remain at the target to allow playback of other content. If so, you will have to manually navigate back to the keypad using the back button shown on the target page.

When this option is selected, you can also set a delay using "Linger after Play". The locator will wait this long once playback ends before closing, giving you some time to play other content at that location. This is useful if you have multiple playable items.

Geolocation Performance

Controls the accuracy versus speed of GPS positioning. Applies only when Geolocation is enabled. Select "Most accurate" only when you need zones to be as small as possible or located in close proximity. The other options result in faster operation, with "Balanced" generally providing a reasonable compromise between accuracy and the time required.

◆ **NOTE:** GPS position accuracy varies among phone brands and models. Make sure to test your system with a number of representative models, verifying sufficient precision and speed.

Optional Binding

If specified, the property will be set when a spot is located. Likewise, if the property is changed by other means, the locator will locate accordingly. The value being set or acted upon depends on the bound property's value type:

- **Boolean** will be set to true while located, and false while at the idle locator waiting for something to be located. Setting the property by other means has no effect.
- **Number** will be set to the location ID of the located spot (if using location ID), or to -1 if not using location ID (e.g., when locating a spot by name). When set by other means, the locator will attempt to locate the spot with that location ID.
- **String** will be set to the name of the located spot (or its dot-separated path, if inside a group). When set by other means, the Locator will attempt to locate the spot specified by the property, by name.

Using a Block for both Display and Visitor Spots

Content targeting both a Display and Visitor spots (by means of a Locator) must be designed with both theses screen sizes in mind. A Tag Selector (or a Synchronizer) is often used as the topmost block. The first child block is set to appear on the Display spot. Subsequent child blocks are tagged with "mobile", making them appear on Visitor spots (a "mobile" tag is added to all Visitor spots by default). See under Synchronizer for more details.

Synchronizer ▸◂

A Synchronizer block synchronizes content, such as video, audio or subtitles, within or across spots. Use cases include:

- A Visitor spot playing audio in sync with a Display spot, for instance using a Locator to connect the spots temporarily for synchronization purposes.
- Two or more Display spots playing synchronized video across those spots, or other content (such as subtitles). These display spots must then be in the same spot group and must be playing the same block (more below under "Synchronizing Video Across Multiple Displays").
- Subtitles playing synchronized to video or audio, on the same spot or on another one (e.g., audio narration appearing as text on visitor's phones for the hearing impaired).

The Synchronizer has the following settings:

- **Synchronization Source** lets you chose to synchronize to a specific property rather than to a Spot in the same group or targeted using a Locator. See under "Sync Source Property" below for more details.
- **Play Automatically** starts content playback when the Synchronizer appears. Uncheck this if you want to start playback by other means, such as a button or a task.
- **Auto-play Delay** specifies how long to wait before starting automatic playback. Normally, you want playback to start right away (delay set to zero). Adding a small amount of delay can improve the synchronicity at the start when starting content across multiple Spots.
- **Loop Content** makes media inside the synchronizer play in a loop. Use this checkbox rather than the Loop checkboxes in each media block for proper looping within a Synchronizer.
- **Local Synchronization** should be selected when you intend to synchronize content only *within* a single spot, rather than across separate spots. This is often used with the Subtitle block, but can also be used to synchronize multiple video or audio media within a single spot.
- **Sync Source Property** specifies a time property to synchronize to. This advanced setting, available only when "Property" is selected as the Synchronization Source, provides explicit control over the synchronizer's time source.

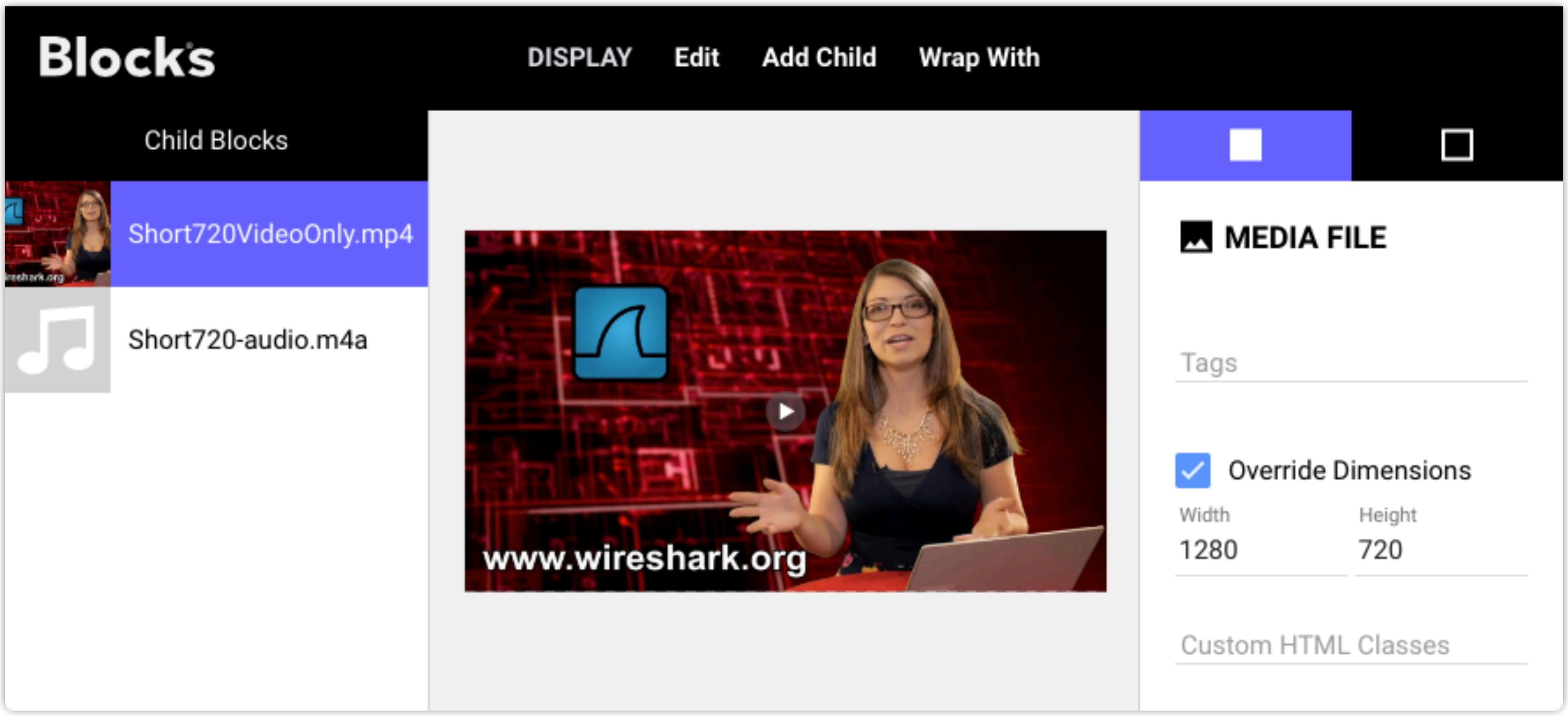
In addition to the synchronization aspect, this block also has similar tag matching capabilities as the Tag Selector, often used to select the proper sound track, or other content, based on the user's language preference.

◆ **NOTE:** The Synchronizer is a premium block type, requiring additional license options in order to be displayed.

To play audio on a Visitor spot in sync with a Display or WATCHOUT Spot, do as follows:

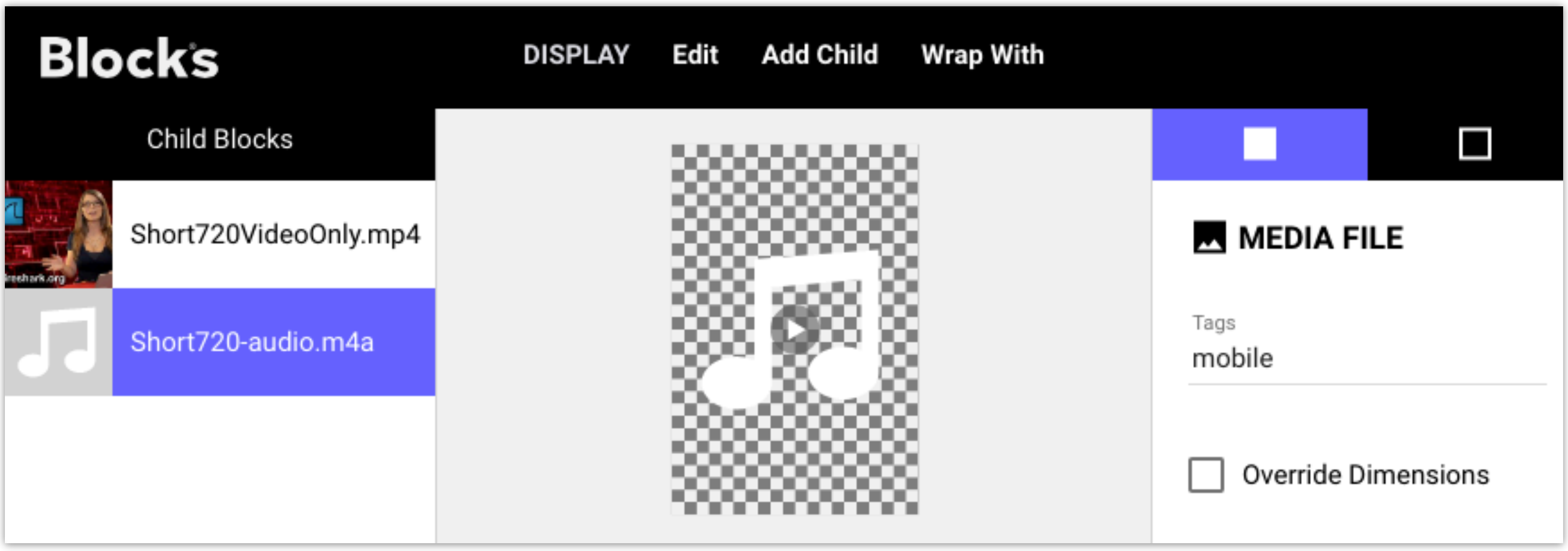
1. Create a Synchronizer block, giving it dimensions appropriate for the mobile device, as in the example shown on the right.
2. Drag the Synchronizer block to a Display or WATCHOUT spot, and give the spot a Location ID.
3. Add the video to appear on the Display spot as the first child block of the Synchronizer, as shown in the illustration below. Override the dimensions of this first child to match the Display spot. (This step doesn't apply when using a WATCHOUT spot.)
4. Add one or more audio media blocks as children of the Synchronizer, assigning tags as appropriate (shown in the second illustration below).
5. Associate a Locator with the Visitor spot.
6. Use the numeric keypad on the mobile device to enter the ID of the Display spot.

The audio will play through the mobile device in sync with the video. To have visual content appearing on the mobile device along with the audio, first add a Composition, then add the sound and other content inside of it.



Override Dimensions: ✓
Width 1280
Height 720

Add a video as the first child of the Synchronizer, when used with a Display spot. Override the dimensions of this child block to match the target Display spot.



Tags: mobile
Override Dimensions: ☐

Add one or more audio blocks to the Synchronizer, assigning tags, such as the mobile tag as well as any language tags, if desired.

Synchronizing Video Across Multiple Displays

You can also use a Synchronizer to synchronize video playback across multiple displays. These videos, which should all have the same duration, may be smaller portions of a high resolution video, cropped to match the arrangement of the displays. Display Spots that are to be synchronized must be in the same Spot Group. You can designate one of them as the "Preferred Synchronization Master" using the checkbox found under the Display Spot's "Advanced" settings tab.

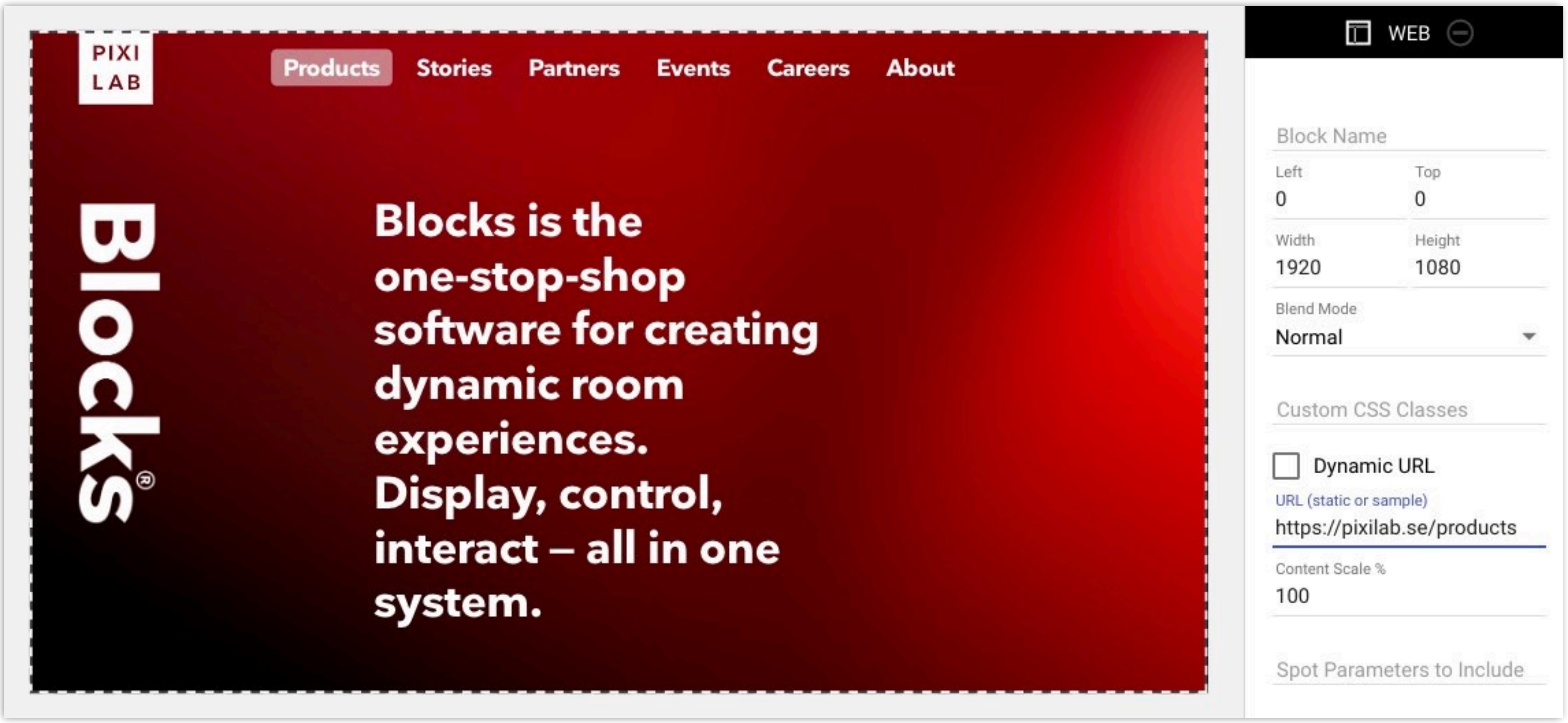
Use tags to associate the video with its display by assigning the same tag to the Display Spot and to the corresponding child block of the synchronizer. For three Display Spots mounted side by side, you may use the tags *left*, *center* and *right*.

Note that you don't need to add a Tag Selector block for this to work, as the Synchronizer also has the same tag selecting ability.

◆ The precision achieved will vary depending on the type of playback devices used as well as the duration of the video. The best precision will be achieved when using PIXILAB Player for fixed display spots and Android devices for mobile devices.

Web

Add a Web block to show a website. This can be a top level block, or a child block inside a Composition, Book, Slideshow or other container block.


PIXI LAB
Products
Stories
Partners
Events
Careers
About
Blocks
Blocks is the one-stop-shop software for creating dynamic room experiences. Display, control, interact – all in one system.
WEB

Block Name
Left: 0
Top: 0
Width: 1920
Height: 1080
Blend Mode: Normal
Custom CSS Classes
☐ Dynamic URL
URL (static or sample): https://pixilab.se/products
Content Scale %: 100
Spot Parameters to Include

◆ Keep in mind that websites may take a while to load. When used in a Slideshow, the web block will begin to load while displaying the previous block. Ensure that the previous block remains on screen long enough for the upcoming web block to load before showing it.

Settings

- **Dynamic URL** lets you specify the URL by binding it to a system property or spot parameter, allowing this to be controlled dynamically rather than always showing the same website.

- **URL** specifies the address to the web page to show. In general, you can copy and paste this from the URL field in a browser to make the same page appear in a web block (see "Limitations" below for some cases when that may not work). When using Dynamic URL, this controls the URL to show in the editor, while the showtime URL will be taken from the bound property.

- **Content Scale** scales the web page by the specified percentage, which is sometimes required to show the web page properly, or with the desired font size. Adjust as needed.

- **Sandbox Website** prevents website navigation away from Blocks, along with some other restrictions. Technically, this is done by applying a "sandbox" attribute to the external website. Does not apply if the URL specifies a PDF document.

- **Spot Parameters to Include** optionally specifies a number of comma separated spot parameters to be appended to the URL as query parameters. This is occasionally useful with web blocks created specifically to be used in Blocks, allowing you to pass parameters to the target web page.

◆ **NOTE:** A block-relative URL, beginning with "~/" (such as *~/custom.html*), may be used if the web block is stored in the block's folder on the server. This is convenient custom web content designed specifically to be used by a particular block, since it will then become part of that bock when exported/imported. This is similar to how a block-relative CSS file can be used.

Restricted websites

Some websites are not compatible with the Web block. For instance, if a website requires log-in in order to work, it will not automatically appear in a web block. In this case, check with the website's owner if it supports other means of authorization, such as a token passed as part of the URL. A website may indicate that it shouldn't be embedded by passing the "X-Frame-Option". You may be able to work around this using browser extensions such as xframe-assassin or iframe-allow for Chrome.

Finally, should you for some reason run your spot using a secure (https) connection, web sites using the plain HTTP protocol can not be shown – only sites that in their turn also use HTTPS. Running Blocks over a secure connection requires a certificate and specific server configuration, which is beyond the scope of this document. Contact PIXILAB for further details if you need to do so.

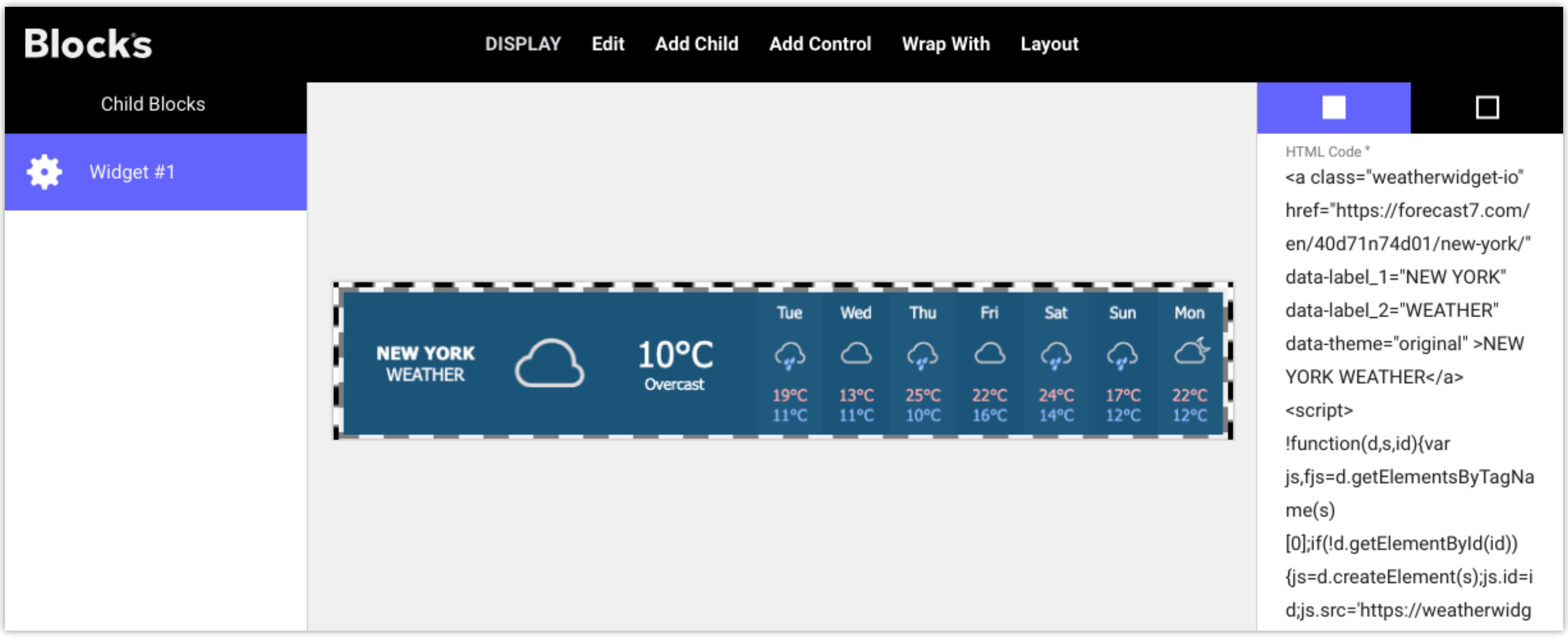
Showing PDF files

Most web browsers have some PDF rendering capabilities. Hence, you can use a Web Block to view a PDF by entering a URL to that PDF. Depending on which playback environment you're targeting, you may be able to append additional options to the URL. For example, to hide toolbars and increase the zoom factor (scale), enter a URL like this (assuming PIXILAB Player or other Chrome-based environment):

```
http://pixi.guide/public/pdf/broschure.pdf#toolbar=0&zoom=200
```

Widget

This block type is similar to the Web block described above. However, rather than a URL pointing to some web content, the Widget block hosts the content inside itself. This method is commonly used to display dynamic content, such as weather maps or clocks.



```
<a class="weatherwidget-io"
href="https://forecast7.com/
en/40d71n74d01/new-york/"
data-label_1="NEW YORK"
data-label_2="WEATHER"
data-theme="original" >NEW
YORK WEATHER</a>
<script>
!function(d,s,id){var
js,fjs=d.getElementsByTagNa
me(s)
[0];if(!d.getElementById(id))
{js=d.createElement(s);js.id=i
d;js.src='https://weatherwidg
```

The weather widget shown above was obtained from the following website.

```
https://weatherwidget.io
```

The widget code obtained from that website was then pasted into the "HTML Code" field, as shown above. Learn more about how to use widgets in the Blocks wiki at https://pixilab.se/docs/blocks/signage#using_widgets.

Reference

Use a Reference block to bring in an entire top-level block as a child of another block. This is particularly useful for content that will appear in multiple places. Instead of copying the same content or block to all those places, do as follows:

1. Add the shared content to the Display page as a top-level block of its own (e.g., as a Media File or Composition).
2. Incorporate that top-level block where needed using a Reference block, specifying the desired block as the "Referenced group/block".

Sharing a single instance of commonly used content in this way makes it easier to update that content in one place, should the need arise, rather than tracking down every place where a copy of the content appears and replace it there.

Select Target Block
The name of the block brought in may be specified explicitly or obtained dynamically through a system property.

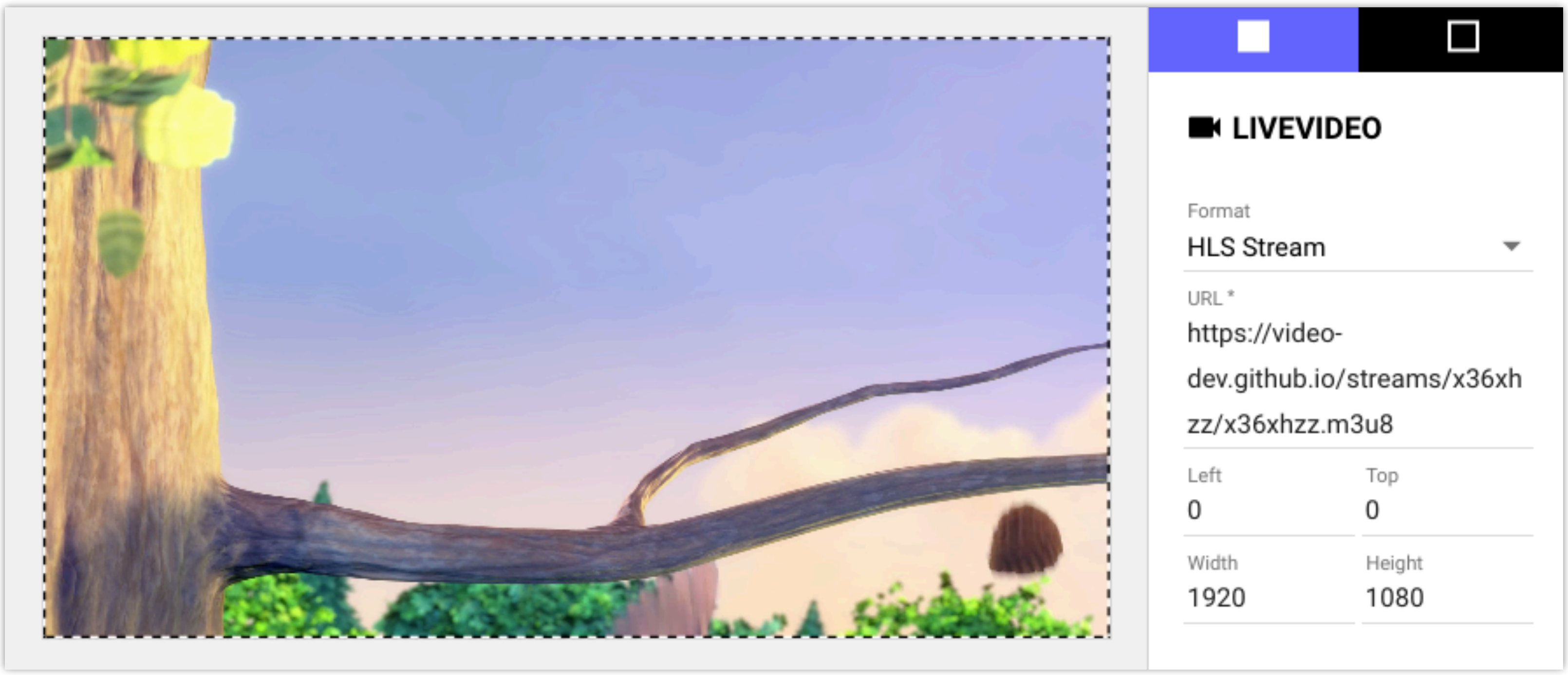
- **By group/block Name**. Selects the desired block using the "Target Block" picker.
- **By Property Value**. The name is specified by the associated system property.

The latter of these two options can, for instance, be used to make one spot show the same block as another spot, perhaps as a small preview window. This is done by binding to the playingBlock property of the other spot.

Live Video

Use a Live Video block to display streaming video, originating from a camera, streaming box, web service or other source. Similar to the Web block described above, you need a URL to the video stream in most cases.

◆ **NOTE:** This is a premium block type, requiring additional license options in order to be displayed.


LIVEVIDEO

Format
HLS Stream

URL *
https://video-dev.github.io/streams/x36xhzz/x36xhzz.m3u8

Left
0

Top
0

Width
1920

Height
1080

Live Video Formats and Settings

Select the expected type of video stream under Format:

- **RTSP stream**. Includes audio. Low latency (generally 0.5 second or less). Supported by most network cameras and streaming boxes. Compatible with most browsers/players.
- **RTMP stream**. Includes audio. Low latency (generally 0.5 second or less). Supported by most streaming boxes. Compatible with most browsers/players. Note that some streaming solutions require the server (here Blocks) to listen for the stream in

order to connect, which means that the stream must be displayed in order for the connection to work. Other streaming solutions (such as the popular ATEM Mini Pro) are more flexible, retrying the connection automatically until it succeeds.

- **HLS stream**. Includes audio. Latency is significant (3 to 10 seconds, or more). Supported by some streaming boxes and most web services. Compatible with most browsers/players.
- **MJPEG stream**. Low latency. Supported by many video cameras and streaming boxes. Audio not included. Compatible with most browsers/players.
- **JPEG polling**. Reasonably low delay (less than 1 second). Generally not full frame-rate. Audio not included. Supported by some video cameras. Compatible with all browsers/players.
- **Local Video In**. Video is provided through a local capture adapter or webcam.Very low latency (a few frames). Useful, for example, to feed a live PowerPoint presentation into Blocks, by connecting the laptop running PowerPoint to a HDMI-to-USB adapter plugged into the USB port of a PIXILAB Player. The adapter must support the "USB video device class" standard, sometimes referred to as "UVC", "driverless" or "class compliant". If there's more than one video input source, select which one to use by entering its number in the **Device Index** field, starting from number 1. Device numbers may shift if such devices are added, removed or connected to other ports on the player.
    - ◆ Local Video In may require specific settings to automatically grant access to the video device. PIXILAB Player is preconfigured with all relevant settings. If using a regular web browser, check under the browser's advanced settings.
- **Internal Video Block**. Video is taken from another visible Video block, essentially duplicating it. Specify the block path to that Video block as the source. This is useful for players with multiple displays attached, allowing the same video to efficiently appear on those displays, possibly with different transformations/behaviors.

Stream Includes Audio

Select this if audio is desired, or if the source provides audio.

- ◆ For the RTSP, RTMP and SRT formats, this setting *must* match the source stream in terms of audio availability. A mismatch here may cause the stream to fail.

Use TCP Transport

Applies to RTSP format only. Normally, RTSP uses UDP to transfer the stream from the camera/encoder to Blocks. In some networks, TCP may be required for a reliable connection. Try this option if the normal UDP method doesn't work reliably.

Source Stream Timeout

Specifies how long to wait for the stream to respond before attempting to reload the stream, in minutes and seconds.

Dynamic URL

If selected, the URL to be used for the stream is obtained from a bound property rather than being specified fixed. This allows you to control the URL to be used by other means or to switch URL on the fly.

Optimize Stream for

The RTSP and RTMP streaming formats can provide reasonably low latency. However, attempting to minimize the latency may result in video dropping out if the source data doesn't arrive on time, since this uses a smaller data buffer.

- **Lowest Latency** minimizes the delay, with the risk of hick-ups or temporarily dropped video if data is delayed.
- **Balanced** provides a good compromise between latency and reliability, with a buffering time of about a seconds.
- **Best Reliability** keeps a larger data buffer, thereby reducing the risk of dropped video at the cost of higher latency.

Playback Start

Controls whether buffering of a network stream happens before playback begins or in parallell with playback.

- **As Soon as Possible** starts playback as soon as sufficient data has been received.

- **Time Accurate** delays playback until the desired amount of data has been buffered. This may result in better synchronicity across displays showing the same live source.

Subtitle

Use this block to render subtitles or closed captioning text, possibly synchronized to video or audio. Blocks accepts subtitles files based on the the SRT or VTT standard file formats. To synchronize subtitles to video, place the video along with the subtitle inside a Synchronizer. Use a Tag Selector to select different subtitles based on language selection or similar.

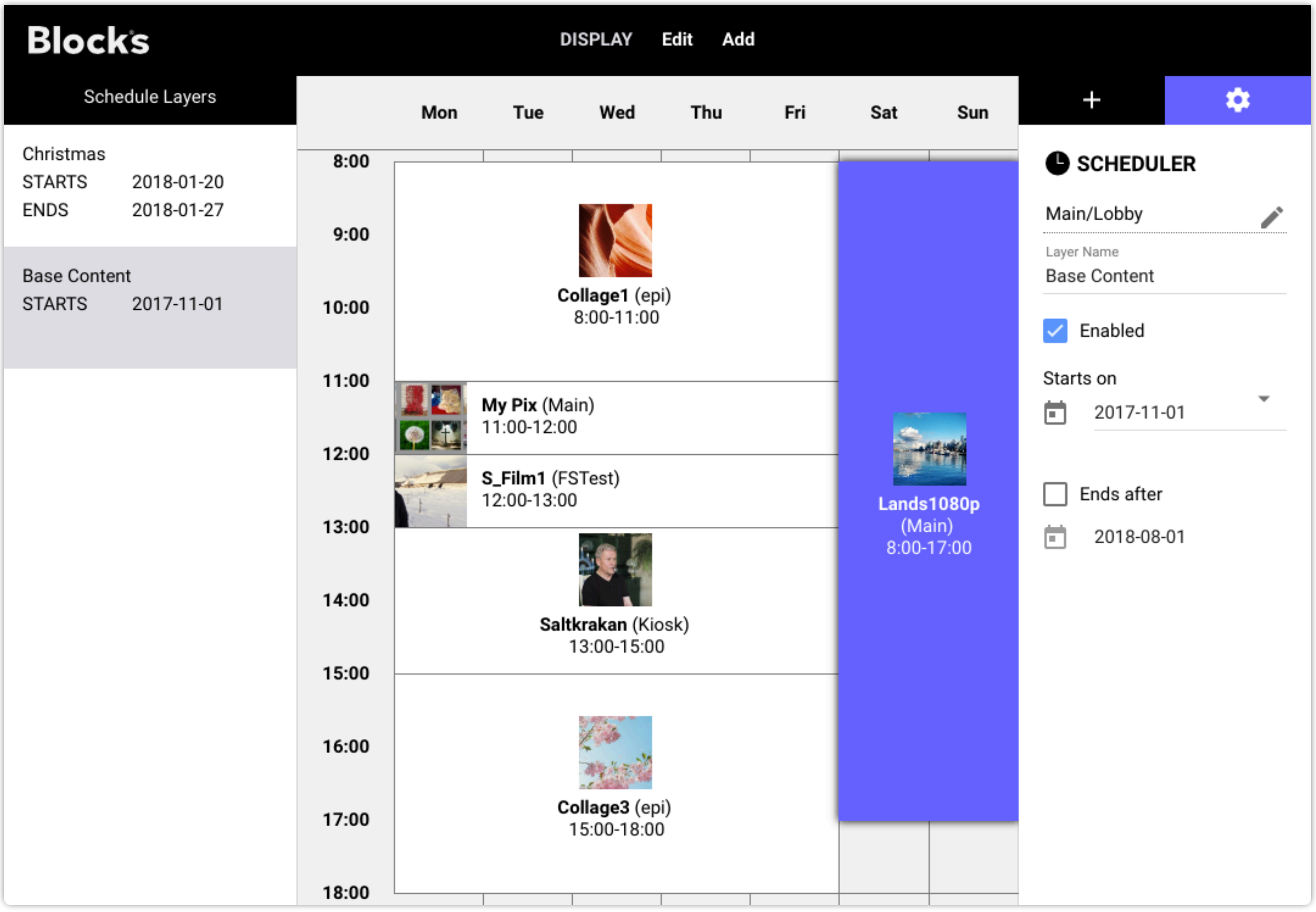
Drag a subtitle file to the box titled "Drop Subtitle File here," shown on the right when a subtitle block is selected. Alternatively, load the subtitle data from the URL specified by a bound property for added flexibility, similar to the Media URL block type.

If the video and subtitles play on the same Spot, then select "Locally Synchronized" for the subtitle. Uncheck this checkbox to synchronize subtitles across spots.

Learn more about how to use subtitles in Blocks here: https://pixilab.se/docs/blocks/app-note/subtitles

Schedule

This block type allows you to show other blocks at predetermined times or dates. A Schedule is assigned to a Spot like any other block. However, a Schedule can not be used inside other blocks; only as a top level block.



Christmas
STARTS 2018-01-20
ENDS 2018-01-27

Base Content
STARTS 2017-11-01

Collage1 (epi) 8:00-11:00
My Pix (Main) 11:00-12:00
S_Film1 (FSTest) 12:00-13:00
Saltkrakan (Kiosk) 13:00-15:00
Collage3 (epi) 15:00-18:00
Lands1080p (Main) 8:00-17:00

Main/Lobby
Layer Name: Base Content
[✓] Enabled
Starts on: 2017-11-01
[ ] Ends after: 2018-08-01

Different content scheduled throughout the day, with other content playing over the weekend.

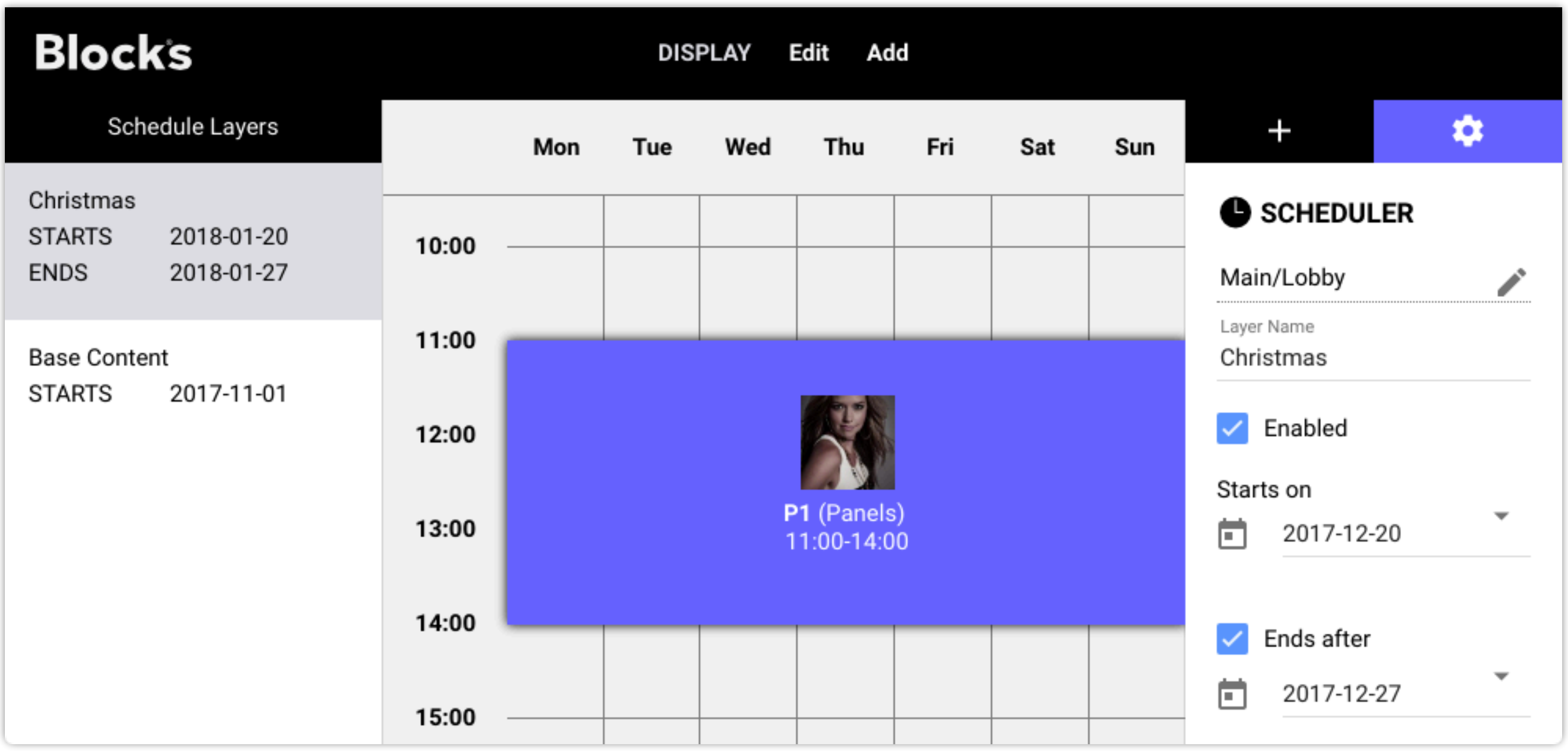
To add content to a schedule, click the "plus" button in the top right corner. Locate the desired block and drag it onto the schedule. Blocks can be filtered on their enclosing group name, making them easier to find if you have lots of blocks in your system.

Once a block has been added to the schedule, move or resize it to determine when to appear or enter its starting/ending time.

Schedule Layers

Content added to a schedule is arranged as layers. Initially, there's just a single layer. To add a Layer, choose "Layer" on the Add menu. A layer can be disabled by unchecking the "Enabled" checkbox. Disabled layers will not play. Likewise, you can constrain the dates the layer will play using "Starts on" and "Ends after". Uncheck "Ends after" to not specify and end date.

Use additional layers to schedule alternative content, while keeping the regular content in place. The system will play the content from on the topmost, enabled layer that is within its date range and has content for the current time and weekday. If there's no content matching those criteria on the topmost layer, the layer below will be considered, and so on.



Christmas
STARTS 2018-01-20
ENDS 2018-01-27

Base Content
STARTS 2017-11-01

Main/Lobby

Layer Name
Christmas

Enabled ✓

Starts on
2017-12-20

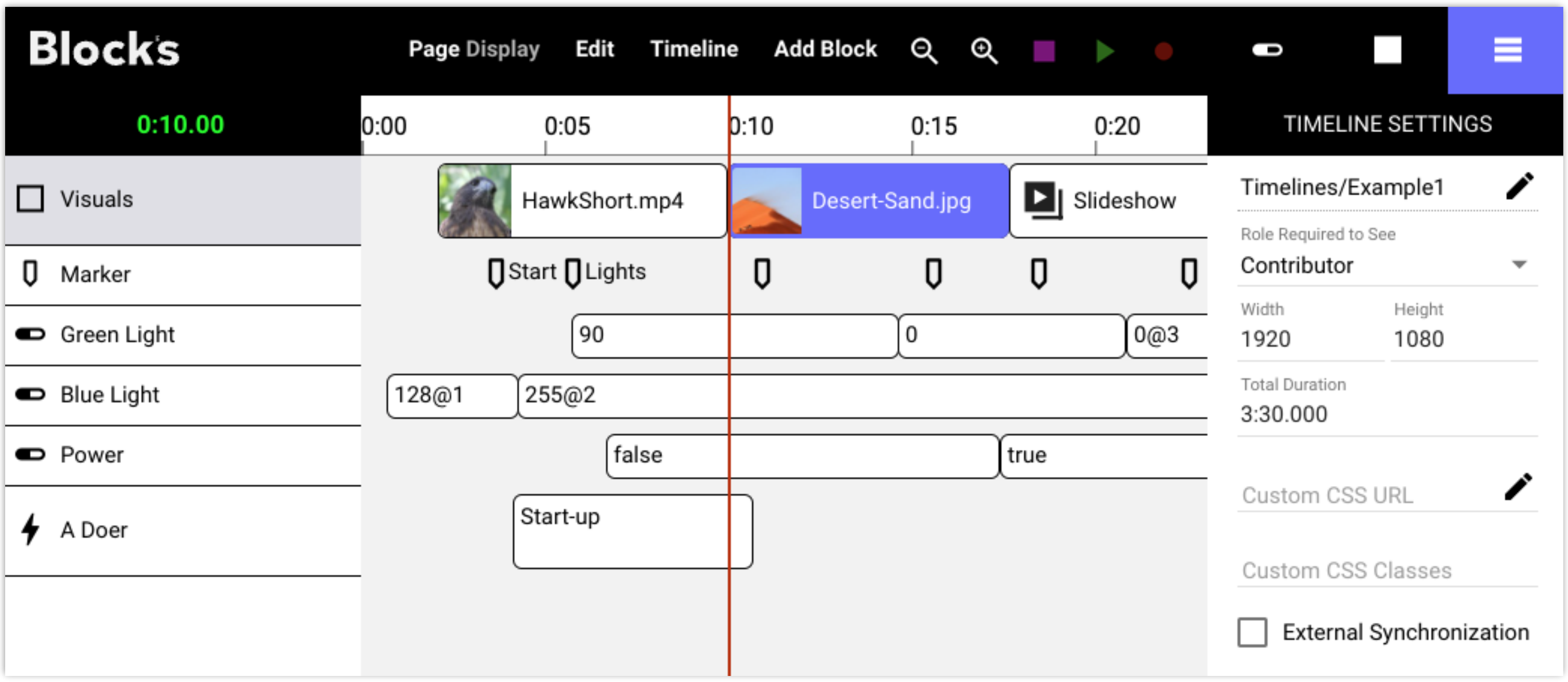
Ends after ✓
2017-12-27

The single block on the Christmas layer will play 11:00 to 14:00 between the specified dates instead of blocks on the Base layer. The Base layer content will continue to play at other times.

6. TIMELINE

The Timeline block, new in version 7, lets you program Blocks in a time-linear way. It's suitable when the desired actions are spread out over time, always performed in the same manner. The Timeline combines display aspects, based on content such as video and images, with control system functions, similar to what you can do with Tasks and Controls such as Buttons or Sliders.

◆ **NOTE:** This is a premium block type, requiring an additional license option.



Timelines/Example1

Role Required to See: Contributor

Width: 1920 Height: 1080

Total Duration: 3:30.000

Custom CSS URL

Custom CSS Classes

External Synchronization

Timeline with Blocks, Markers, three Property tracks and a Do track. Settings for the timeline, selected track/cue and property values appear to the right under their respective tab.

A Timeline is created, edited and used as other block types. For instance, it can be assigned to a Spot, making its visuals (blocks) appear on that Spot. But it also has a life of its own, independently of whether it currently appears on a Spot. In that respect, it is similar to a Task or the Schedule block. However, while the Schedule block performs coarse scheduling of content based on time of day, the Timeline block performs fine-grained scheduling and synchronization of blocks and control functions that can be started, positioned and stopped at will from a Button ot Task.

Timeline Settings

Most Timeline Settings are similar to those available for other block types, with the following additions:

- **Width and Height** governs dimensions of content. This can then be overridden per block track, if needed.
- **Total Duration** indicates the length of the timeline. It must be large enough to encompass all cues. Adding a cue at the end of the timeline automatically extends it.
- **Play Automatically** starts timeline playback whenever it is applied to a Spot.
- **Loop** makes the timeline restart automatically from its beginning whenever played to its total duration. If not selected, the timeline stops when reaching its end.
- **External Synchronization** lets you choose an external time source to drive this timeline, as well as an offset relative to that time source. This time source can be any other object capable of providing time data, such as a Display Spot playing a video, an external SMPTE timecode feed, a media server or any device with a driver that exposes a time property.

Tracks and Cues

A Timeline consists of a number of horizontal tracks of various kinds. Add tracks using the Timeline menu. To edit a track's settings, select its track header to the left to show the Track Settings pane to the right.

Reordering Tracks

When adding a new track, it will be placed above any currently selected track of the same kind. To move a track up or down the timeline (or to another timeline):

1. Select the track by clicking its track header.
2. Choose Cut on the Edit menu.
3. Select the desired target location by clicking its track header.
4. Choose Paste.

◆ **NOTE:** Tracks of the same kind are always grouped together on the timeline.

Block Track

Use a Block Track to present visual content on Spots. Add a Block track using the corresponding command on the Timeline menu, then add content to it using the Block menu, or simply drag a media file to the track. For more complex block types, such as Composition or Slideshow, double-click the cue corresponding to the block to edit its content.

Override Dimensions lets you manage content of varying aspect ratios or dimensions, overriding the general dimensions of the timeline.

Snap to Cues makes cues on other tracks snap to the beginning of cues on this track. This is turned on by default for the marker track, but may be enabled for all track types.

Default Transition sets the transition used between cues on this track, unless overridden for a cue.

Presenting a Timeline on a Spot

To present a timeline containing blocks on a Spot, drag the Timeline block to the Spot. Alternatively, use buttons, tasks or the Block Scheduler to apply it dynamically. A timeline can appear on multiple Spots. Note, however, that all spots showing that timeline will run according to the timeline's time position.

Using Multiple Block Tracks

A Timeline can have multiple block tracks. By default, only the first track appears on a Spot. Use Tags to make the desired track appear on each spot. This allows you to synchronize content across multiple displays, for instance to make images or video transition at the same time over several displays. Specify a tag (or a set of tags) for the Block Track and then set the same tag on the Spot, for example using the Spot's "Fixed Tags" setting. In this way, the timeline takes on similar functionality as the Tag Selector for showing the desired block track on the right Spot.

For instance, if you have three displays that should play content originating from the same timeline, tag the Display Spots with the tags *left*, *center* and *right* respectively. Then enter the same tag name for each of the three Block Tracks by selecting the track header and entering the tag name in the Track Settings's "Tags" field.

You can override the dimensions for a block track. This is useful if the timeline contains content of varying aspect ratios, for instance targeting a mix of horizontal and vertical displays.

Applying Transitions

Similar to the Slideshow block, a block track defines a default transition in its Track Settings. This transition is applied between blocks on this track when rendered onto a Spot. If desired, you can override this default transition on a block-by-block basis by selecting the block cue and specifying another transition under its Cue Settings.

Block Cue

Add a Block to a block track to create a cue. Alternatively, a Media File cue can be created by dragging an image or video from your computer onto a block track. Cues for content with an inherent duration, such as a video, show a gray bar along the bottom indicating this duration. **Desired Duration** lets you override this inherent duration or specify a desired duration for other block cues. If snapping is enabled, the following cue will snap to the end of the indicated duration. If the following cue has a transition, it snaps at the end of that transition.

A Block cue has the same settings as blocks in other contexts (such as in a Slideshow), plus a time position. If you have more than one Block track, select the track before adding the cue. Cues can be moved between tracks or timelines using the Cut and Paste commands of the Edit menu.

Marker Track

The main purpose of the Marker track is to let you drop markers at interesting points along the timeline. These markers can be used as reference points for adding other cues. The "Snap to Cues" setting is enabled by default for a Marker track, making other cues snap to markers as they are dragged along their tracks. This speeds up temporal cue alignment based on markers. This setting can also be enabled for other track types.

Marker Cue

To add a marker, choose "Add Marker Cue" on the Timeline menu, or just press M on the keyboard.

- ◆ **HINT:** This can be done while playing the timeline, allowing you to use this method to then time other cues to audio.

Name markers to make them easier to locate or for use as jump targets for *gotoMarker* functions.

Jumping to Markers

Naming a marker cue as a function key (e.g. "F4") lets you jump to that marker by pressing that key. A Do cue (on the same timeline, on another timeline or in a task) can also be used to jump to a named marker.

Property Track

A Property Track controls a single property, such as the intensity of a lighting fixture or the state of a relay. This is similar to how you can control a property manually by binding it to a Button. Add one track per property you want to control from the timeline. Specify the property to control in Track Settings, shown on the right while the track heading is selected.

For properties with a known value range (often normalized as 0…1), you can specify how to view and edit the value on the "Show Value as" dropdown menu. For instance, if you're controlling lighting channels, you may prefer to specify the value as 0…255, according to the DMX-512 value convention, rather than as 0…1.

- ◆ **NOTE:** This setting controls only how the value is shown in the user interface. The internal value remains the same.

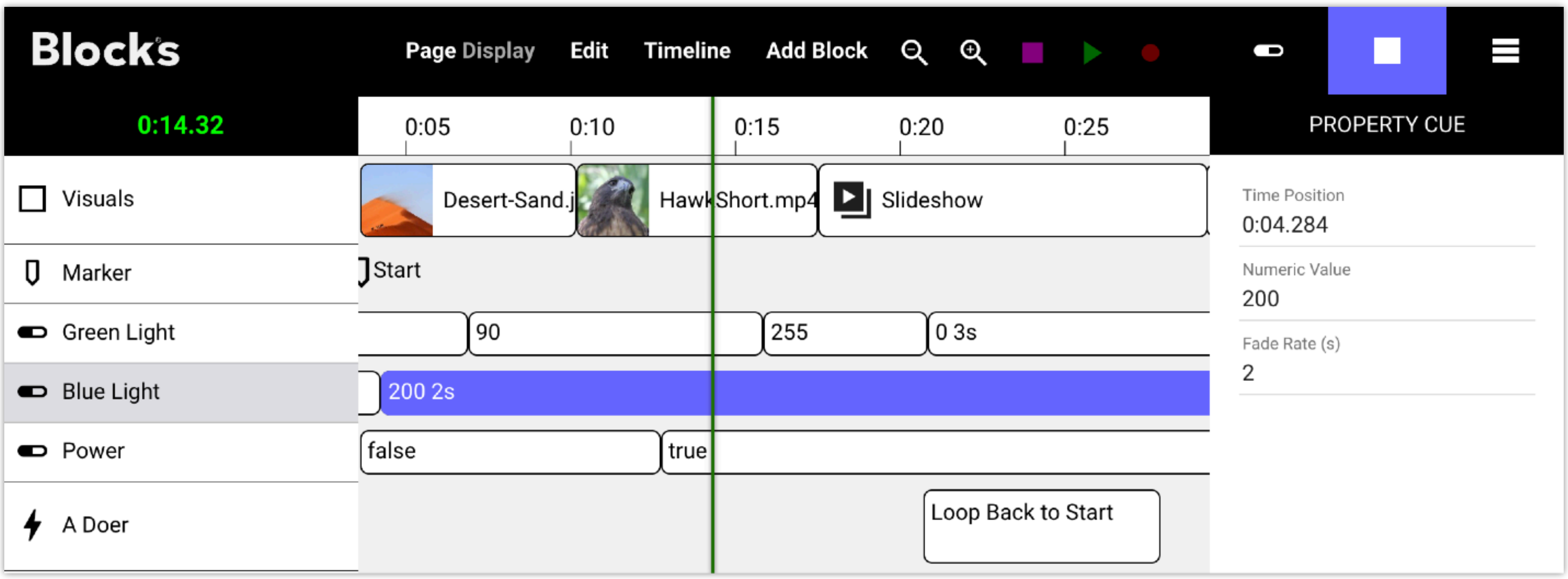
You can see the current value of each property by selecting the "Property Values" tab in the inspector. This also allows you to change the value of the property.

- ◆ **HINT:** You can disable a Property track by unchecking its Enabled checkbox. While disabled, cues on this track have no effect. While disabled, you can't use the Auto Record mode and the controls in the Property Values pane to program cues on this track.

Property Cue

A Property Cue specify what value a property should take at that point along the timeline. Select a cue to edit its settings.

Property Cue settings edited in the settings pane on the right.



Time Position
0:04.284

Numeric Value
200

Fade Rate (s)
2

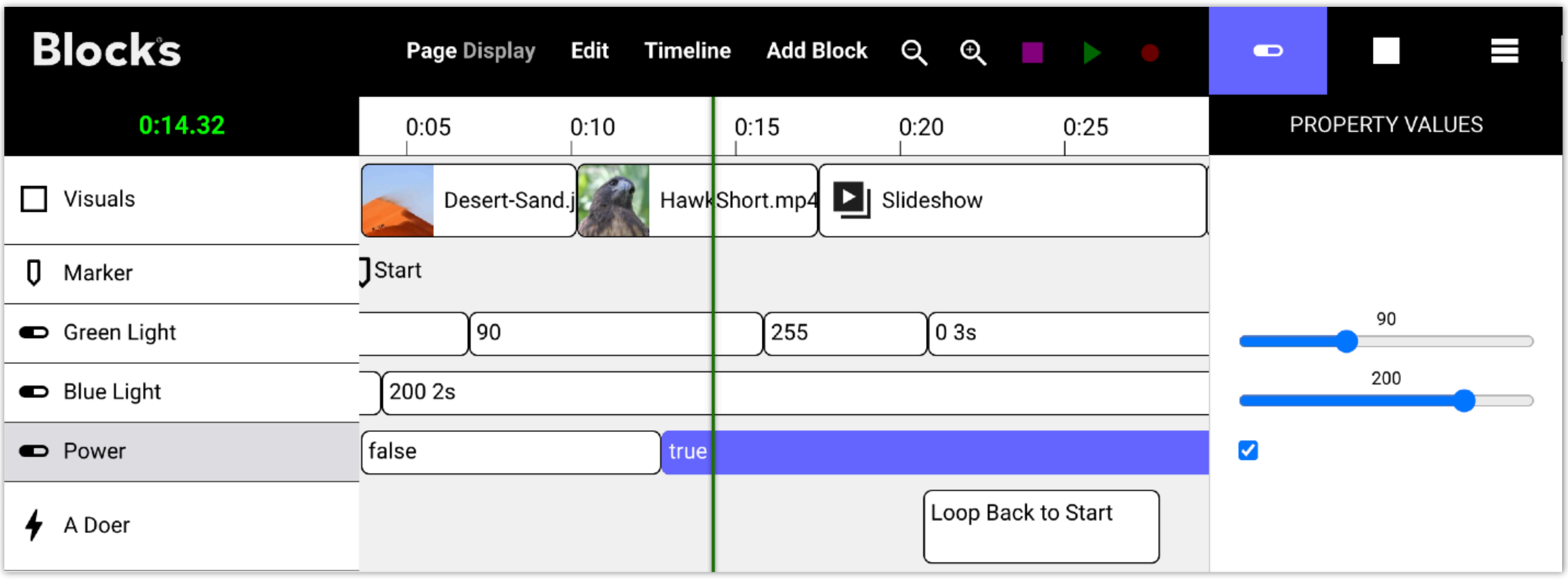
Settings available vary with the property type:

- **Numeric** property has a value field. If the property value has a sibling *fadeTo* callable (as all DMX-512 channels do, for example), a "Fade Rate" setting also appears. Leave "Fade Rate" unspecified to use the default fade rate, if any.
- **String** property has a text field for entering the new text.
- **Boolean** property has a checkbox for setting the desired state.
- **Enumerated** properties have a dropdown list of available values

  ◆ **HINT:** As an alternative to editing the value of a cue as discussed here, you can change its value in the Property Value tab. See below for more details.

Property Values

The current value of properties are shown under the Property Values tab. You can change the value of most properties here.



Property values appear and can be manipulated in the settings pane on the right.

A new cue added to a Property track takes on the value shown. Furthermore, by enabling the "Auto-Record Property Changes" mode (see under Timeline Commands below), property cues will be updated/added as you change the value here.

◆ **NOTE:** Only numeric properties with a known range value will show a slider. Unbounded numeric values will appear as a value field, without the slider.

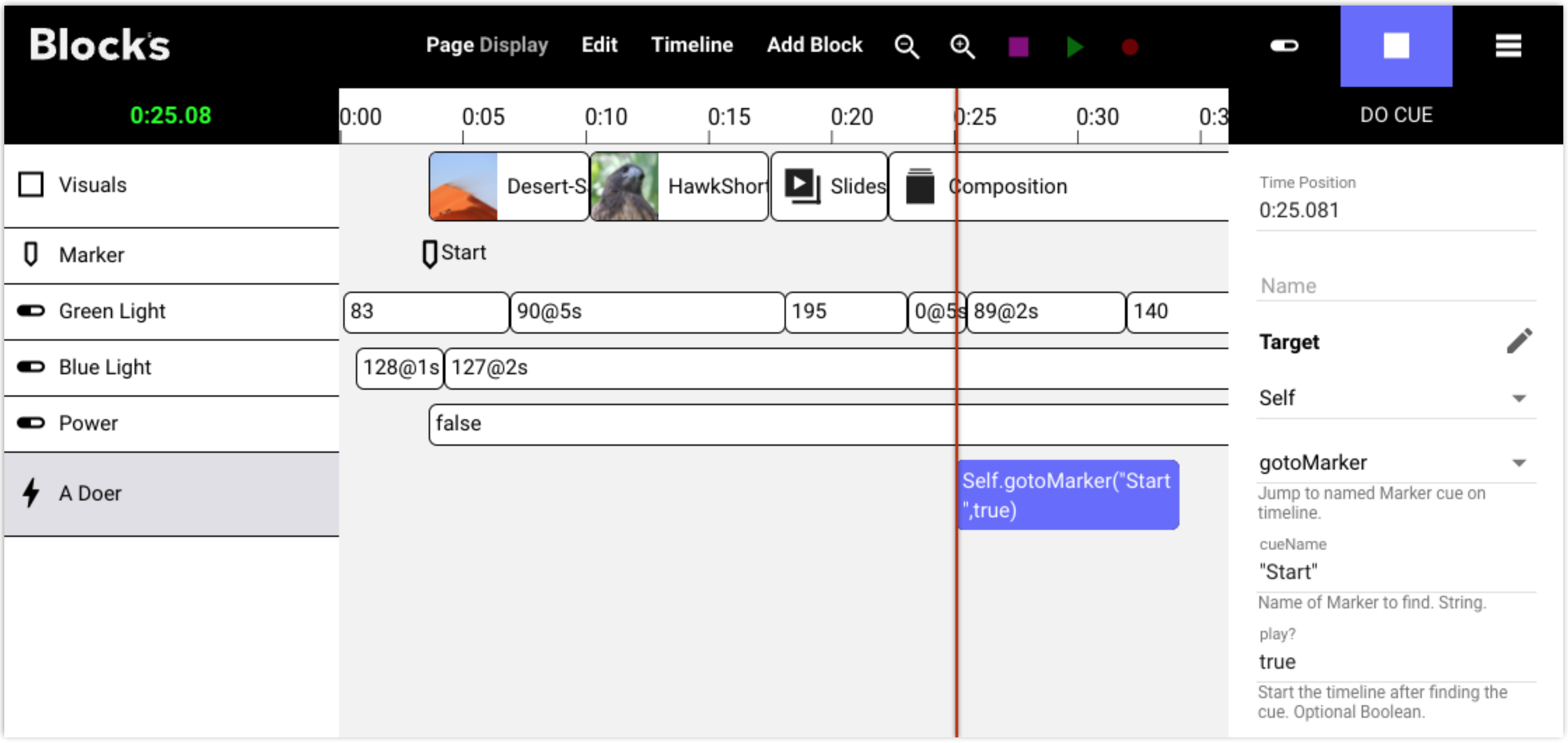
Do Track and Cues

Cues on a Do track are similar to *do* statements in Tasks in that they can set any property value or call any callable function in Blocks.

◆ **IMPORTANT:** While Property cues perform as you jump around on the timeline, Do cues perform only when you run over them by playing the timeline.

While a Do cue can set the value of any property, it's often a better choice to use a Property track for this purpose. Especially if you need to set the property repeatedly along the timeline, or want the property value to follow suit as you jump around on the timeline.

Do cues have an additional top level property path item named "Self". Use this to apply a cue to the timeline itself, for example to pause the timeline (by setting *Self.playing* to false) or make it jump to a named marker cue.



Time Position
0:25.081

Name

Target
Self

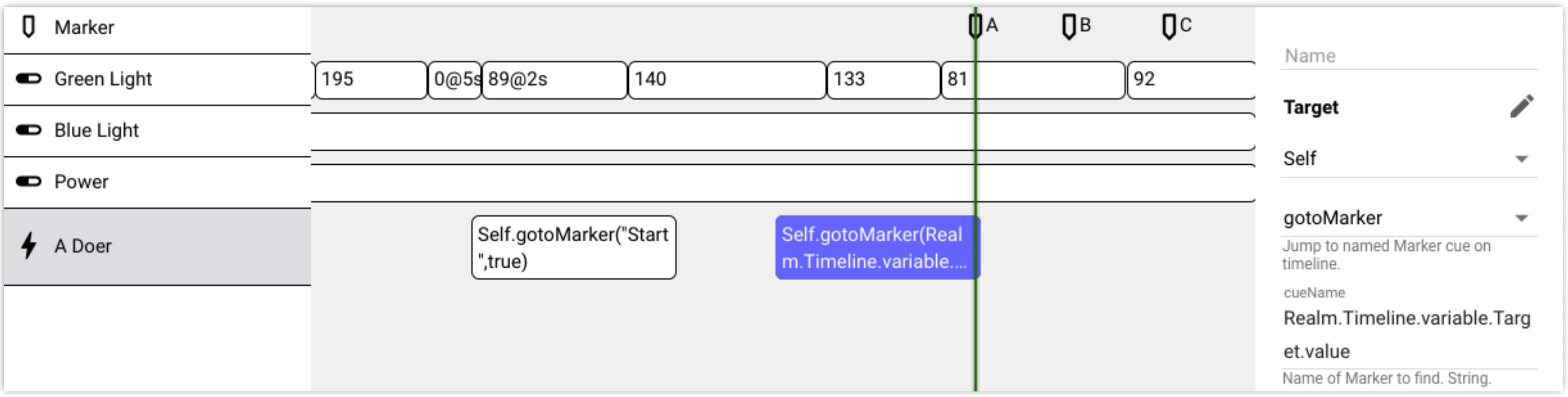
gotoMarker
Jump to named Marker cue on timeline.

cueName
"Start"
Name of Marker to find. String.

play?
true
Start the timeline after finding the cue. Optional Boolean.

A Do cue targeting its enclosing timeline using "Self" to create a loop.

You can use expressions in value fields, for example accessing the value of a property using its property path. This is similar to Task *do* statements. For instance, you can use this to jump to different markers depending on a Realm variable specifying the name of the marker.


Marker
A
B
C
Green Light
195
0@5s
89@2s
140
133
81
92
Blue Light
Power
A Doer
Self.gotoMarker("Start",true)
Self.gotoMarker(Realm.Timeline.variable....

Name

Target
Self

gotoMarker
Jump to named Marker cue on timeline.

cueName
Realm.Timeline.variable.Target.value
Name of Marker to find. String.

A Do cue that jumps to different markers based on the content of a Realm variable.

◆ **HINT:** You can disable a Do track by unchecking its Enabled checkbox. While disabled, cues on this track have no effect.

Timeline Commands

Some Timeline-specific commands are available in the menu bar at the top of the window.

🔍 Zoom in on the timeline, for making fine adjustments to time position of cues. Keyboard shortcut: plus key.

🔍 Zoom out, letting you see more of the timeline. Keyboard shortcut: minus key.

■ Stop and disengage the timeline so it has no effect.

▶ Play the timeline. Click again to pause. Keyboard shortcut: spacebar.

● Enables the "Auto-Record Property Changes" mode. Keyboard shortcut: R.

Auto-Record Property Changes
This mode is activated through the Timeline menu (keyboard shortcut R) or by clicking the "record" symbol in the menu bar. While in this mode, any changes you make in the Property Values tab will automatically update or create a new Property cue accordingly.

Keyboard Navigation
The following keys navigate along the timeline:

- **Plus** zooms in, giving you better time precision when moving cues.
- **Minus** zooms out, viewing more of the timeline.
- **Right** and **Left** arrow keys skip to the next/previous cue.
- **Home** and **End** jumps to either extreme on the timeline. Press Shift to also move the current time position.
- **Page Up** and **Page Down** scrolls forward or reverse. Press Shift to also move the current time position.

Edit Menu

The following commands are specific to the Timeline editor:

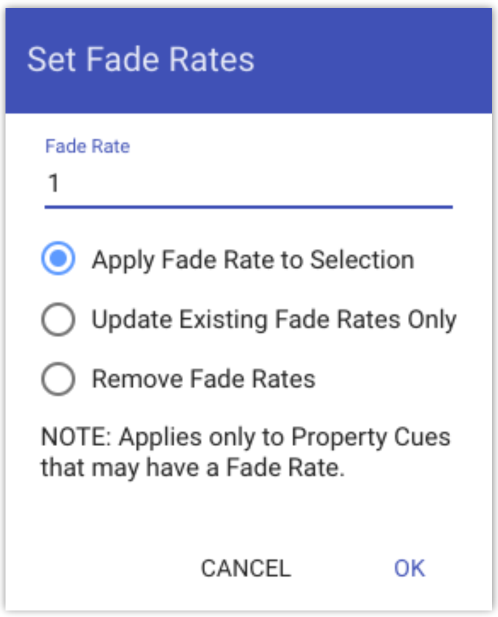
- **Select to End** selects all cues ahead on selected track(s).
- **Select All to End** selects all cues ahead on all tracks.
- **Select None** deselects all cues.
    - ◆ **HINT**: You can also select all cues ahead by Option-clicking a cue. Press Shift as well to add to the selection.

Timeline Menu

- **Add Block Track** adds a new track for holding video, images or other blocks. Use the commands on the Add Block menu to add content to such a track, or simply drag an image or video onto the block track.
- **Add Marker Track** adds a new track for markers. You can use multiple, named, marker tracks for different purposes. If you have multiple tracks, select the desired track before adding a marker. The "Snap to Cues" option is automatically enabled for marker tracks, making other cues snap to cues on this track. You may want to deselect this when using multiple marker tracks.
- **Add Marker Cue** adds a marker. Named markers can be used as targets by the *gotoMarker* function (as shown in the example under Do Tracks and Cues above). Markers can be added while playing the timeline using the keyboard shortcut M. Then use those markers to align events to marked positions of sound track, for example.

- **Add Property Track** adds a new track for controlling a specific property. You must select the target property in the Track Settings pane after adding the track. See Property Track above for more details.
- **Add Property Cue** adds a new cue to the currently selected property track. Keyboard shortcut P. The cue will take on the current value of that property. You can set the desired value prior to adding the cue using either the slider in the Property Values tab or by other means (for example, a slider on a control panel shown on a tablet). Change an existing property cue either manually in the Property Cue settings tab or by using the Auto-Record Property Changes mode described above.
- **Set Fade Rates** lets you change the fade rate of multiple property cues at the same time. More on this below.
- **Add Do Track** adds a new track for holding Do cues.
- **Add Do Cue** adds a cue to the currently selected Do track. See Do Track and Cues above for details.
- **Go to Time** lets you enter the desired time position numerically. Keyboard shortcut G.
- **Scroll to Current Time** reveals the current time position when scrolled out of view. Keyboard shortcut T.
- **Click Jumps to Time** sets the time position of the timeline to the starting position of a cue when selected. Deselect this option to avoid affecting the current time position when selecting cues. Keyboard shortcut J.
- **Snap Cues** causes cues dragged along their tracks to snap to cues on other tracks. This is typically used to snap cues to markers, but can be used to snap to cues on any track by enabling "Snap to My Cues" for that track. Keyboard shortcut S.

Set Fade Rates


Set Fade Rates

Fade Rate: 1

(● Apply Fade Rate to Selection) ○ Update Existing Fade Rates Only ○ Remove Fade Rates

NOTE: Applies only to Property Cues that may have a Fade Rate.

This command lets you set the fade rate of selected property cues in one go. This applies only to property cues having a "Fade Rate" setting, such as those associated with DMX-512 lighting channels and similar. See Property Cue above for more details.

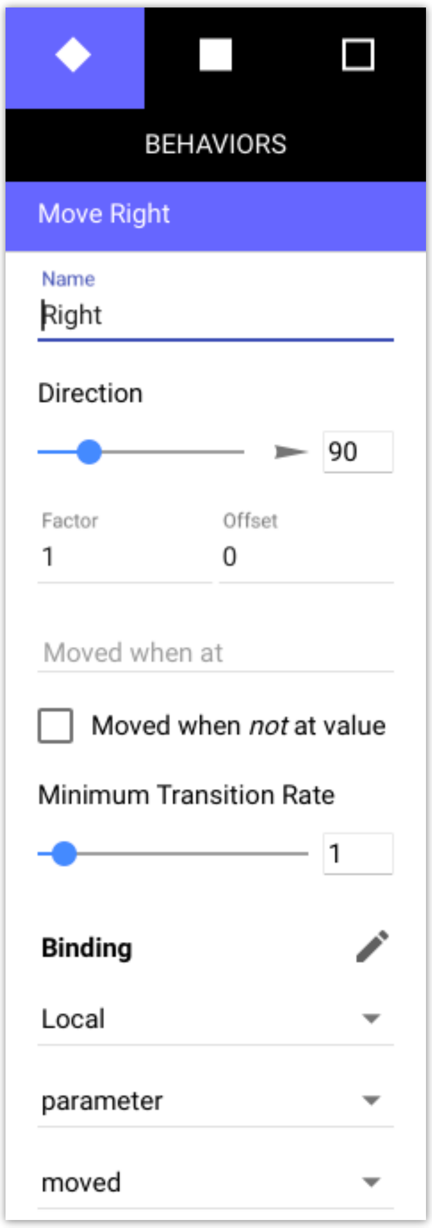
Enter the desired fade rate to be applied, if any, and select the desired mode:

- **Apply Fade Rate to Selection** changes the fade rate of all applicable cues to the rate specified.
- **Update Existing Fade Rates Only** changes the fade rate only of cues that already had a fade rate set.
- **Remove Fade Rates** clears the fade rate of cues that had one set, making the cue use the default fade rate for the property, if any.

7. BEHAVIORS

Behaviors add interactive capabilities to child blocks. Behaviors are added to a child block using the Add Behavior menu. Behaviors are bound to local or system properties. Depending on the behavior selected, the property may either affect the child block or vice versa.

To view any behaviors attached to a block, first select the block in the list to the left in the editor, then click the diamond-shaped symbol in the top right hand corner, as shown in the illustration on the right.



Some behaviors, for instance Reveal, are applied when at a particular property value. Those behaviors behavior have an "…at value" field where you can enter the value to me matched. If this field is left empty, the behavior will be applied when the property's value is non-zero number, non-empty string or the boolean value *true*. If this field contains the literal value *falsey*, this logic is inverted so that the behavior will be applied when the property's value is number 0, empty string, boolean *false*.

You can add multiple behaviors to a block. The example shown on the right has a single, currently selected Move behavior. Additional behaviors appear in the list at the top. A behavior can be named, which can be helpful when applying multiple behaviors of the same type. Click a behavior in this list to view its settings.

You can copy and paste behaviors between blocks:

1. Select the desired behavior(s) in the list.
2. Choose Copy on the Edit menu.
3. Navigate to the destination child block.
4. Choose Paste.

Blocks with behaviors are shown in *italics* in the list of child blocks as well as in hierarchy views.

- ◆ Behaviors apply when displaying the block on a spot. They do not apply in the editor. Use a test spot, set to "Update on Block Change," to view the resulting behavior while editing.

Appears

The bound property will be set to specified value (or *true* if no value specified) when the block appears. This can be used to trigger an action when the block appears on screen, such as when scrolled into view in a Scroller. The threshold value specifies how much of the block that needs to be visible for it to be considered. Set a small value (e.g., 5 percent) to make the behavior take effect when the block is only partially visible, or a large value to make it take effect once the whole block is entirely visible (100 percent).

Disappears

The bound property will be set to specified value (or *true* if no value specified) when the block disappears. This can, for instance, be used to trigger an action when leaving a page in a Book, or when a video is scrolled out of view.

Move

Makes the block change its position on screen according to the bound property.

Direction
Specifies the direction of motion.

Factor and Offset
The block will be moved by the distance resulting from this equation:

> *bound property value * factor + offset*

This is useful, for instance, when dealing with a normalized (0…1) property value. To move the block by 200 pixels over the range of the property, set the *factor* to 200. *Factor* and/or *offset* may be negative.

Moved when at
When bound to a string or boolean property, specify the state of the property when to apply the motion.

Moved when *not* at value
Inverts the motion condition specified by "Moved when at". This is sometimes useful in conjunction with a string property, allowing you to apply the motion while *not* at the specified value.

Minimum Fade Rate
Set to a time, in seconds, to move gradually over the specified time, rather than jumping to the destination position.

Mute

Mute the audio, based on the value of the property binding. Applies to the following block types: Media, Media URL and Live Video.

- ◆ **NOTE:** If no property binding, then mute unconditionally. This is sometimes useful to silence unwanted audio in video. This can also help when targeting browsers that won't auto-play video containing an audio track (even if silent).

Mute while at Value
If specified, audio from this block will be muted while the property is at the specified value. If not specified, the value of the property directly controls the mute function, with a value of 1, true or non-empty string mutes audio.

Play

Makes the block (e.g., a video, audio or a slideshow, or any block containing such blocks) play or pause, based on the value of the property binding. Applies only to blocks that can play, such as video, audio and slideshow.

Play while at Value
If specified, the child block will play while at the specified value, else it will be paused. This is sometimes useful in conjunction with a string property, whose value then governs the playback. If not specified, the value of the property directly controls the state of the block, with a value of 0 or false pausing it and 1 or true playing.

Playing

The bound property will be set to *true* while the block (or any of its children) plays, and set to *false* once playback stops. This can, for example, be used to trigger actions when a particular video starts to play.

Press

Simulates pressing a button, video, camera, etc, based on the value of the property binding. This allows functionality of buttons and other blocks that can be "pressed" to be controlled programmatically

Press while at Value

If specified, the block will be pressed while the property is at the specified value, else it will be released. If not specified, the value of the property directly controls the behavior, with a non-zero value ora boolean *true* causing the block to be pressed.

Reveal

Causes the block to be shown when the property is set. This is particularly useful to reveal a deeply nested block, for instance inside a Book, Slideshow or similar.

For this behavior to have any effect, the block it is attached to must not already be shown. This can be accomplished, for instance, by putting this block on a page in a book that's currently not visible. When the reveal behavior is triggered, the book (or other enclosing blocks) will be navigated to reveal the block. If you merely want to show and hide a block that's already on screen, you can use an Opacity behavior instead, which only controls the opacity of its block without causing any navigation of enclosing blocks..

- ◆ **NOTE:** This behavior does not work inside replicated child blocks.

Reveal when at Value

If specified, the block will be revealed while at the specified value. This is sometimes useful in conjunction with a string property, whose value then governs the appearance. If not specified, the value of the property directly causes the block to appear when set to a *truthy* value, i.e., a value that is not zero, false, undefined or an empty string.

Only if Parent Visible

When selected, the block will not be revealed unless its directly enclosing block is already visible. If not selected, the block will be revealed regardless, even if deeply nested inside enclosing blocks that may or may not be visible already.

Navigate

Applies to blocks that have children that can be navigated to, such as Book, Slideshow, Scroller and Grid. Child navigation will be triggered when the bound property changes. The property may be a number or a string:

- A number will navigate to the index position of the child, where the first child has index 0.
- A string will navigate to the child with the specified name. If the property specifies a relative block path (such as "a/b/c"), navigation will descend to also show the specified children.

Rotate

Makes the block rotate on screen according to the bound property, using a value specified in degrees, with positive values rotating clockwise. If no property binding then rotate by 180 degrees or by the amount specified as Offset

Factor and Offset

The block will be rotated by the number of degrees resulting from this equation:

> *bound property value * factor + offset*

This is useful, for instance, when dealing with a normalized (0...1) property value. To rotate the block 360 degrees over the span of the property, set the *factor* to 360. *Factor* and/or *offset* may be negative.

Applied while at
When bound to a string property, specify the state of the property at which to apply the rotation.

Minimum Fade Rate
Set to a time, in seconds, to rotate gradually over the specified time, rather than jumping to the destination angle.

Scale

Similar to the Rotate behavior described above, but affects the block's size instead. A scaling value greater than 1 makes the block larger and smaller than 1 makes it smaller.

Zoom

Lets you use touch screen pinch gestures or a mouse scroll wheel to enlarge and pan the content, which would typically be an image or a composition with a mix of text and images. When used with images, those need a higher resolution than what's initially required to allow enlargement without noticeable quality loss. Set the "Max Zoom Factor" to roughly match the resolution available when enlarging the content.

- **Mouse Wheel** allows use of the scroll wheel on a mouse to zoom the content.
- **Max Zoom Factor** limits the maximum enlargement that can be applied.

Speed

Affects the playback rate of video and audio. The value of its property binding controls the speed. A value of 1 plays at normal rate. A value greater than 1 plays faster. Select "Preserve Audio Pitch" to keep the audio pitch constant while changing the playback rate. Uncheck this option make the pitch follow the playback rate. **Factor** and **Offset** are useful when you need to use a normalized value (0…1) to control a speed with another span (see the Move behavior for more details). The **Minimum Fade Rate** field lets you change the speed gradually (see the Opacity behavior for details).

◆ **NOTE:** This behavior isn't compatible with some players, such as iOS, where the speed can't be changed gradually. Very fast playback will not work properly on most devices, due to limitations in audio/video decoding performance. Very slow playback will result in choppy video playback.

Volume

Controls the playback volume of its media block. Settings are similar to the Opacity behavior described above.

Classify

Conditionally applies custom CSS classes to the block, based on the value of the property binding. When specifying multiple classes, class names must be separated by a space.

- **Fixed Set of Classes.** Applies the classes specified as "Fixed CSS Classes" when the bound property is true or has the value specified as "Apply at Property Value".
- **Classes from Property.** Applies the class or classes specified by the bound property (which must be a string).

Fixed CSS Classes
The custom class to be applied. This is similar to the Custom CSS Classes that can be applied statically to any block, but here the class is applied only under the condition specified.

Apply at Property Value
If specified, the class will be applied while at the specified value. This is sometimes useful in conjunction with a string property. If not specified, the value of the property causes the class to be applied while at a *truthy* value (i.e., a value that is not zero, false, undefined or an empty string).

Opacity

Shows and hides the block based on the value of the property binding. Hides unconditionally if no property binding.

Visible while at Value
If specified, the block will be visible when at the specified value, else it will be hidden. This is sometimes useful in conjunction with a string property, whose value then governs the visibility. If not specified, the value of the property directly controls the opacity of the block, with a value of 0 or false being hidden and 1 or true being fully visible.

Minimum Fade Rate
Set to a time, in seconds, to fade gradually over the specified time, rather than showing or hiding instantly.

Blur

Blurs the content of the block, either by a *Fixed Value* or based on the value of a property. When bound to a property, a Scale Factor can be applied – particularly useful when bound to a normalized (0…1) value; e.g., by then scaling that normalized value by 10.

Contrast and Brightness

Changes the contrast or brightness of the block, either by a *Fixed Value* or based on the value of a property, with normal contrast/brightness corresponding to a value of 1.

Grayscale and Sepia

Removes colors from the block, rendering it with a grayscale or sepia tone instead based on the value of a property, with the block being unaffected at value 0 and fully grayscale/sepia at 1.

Hue

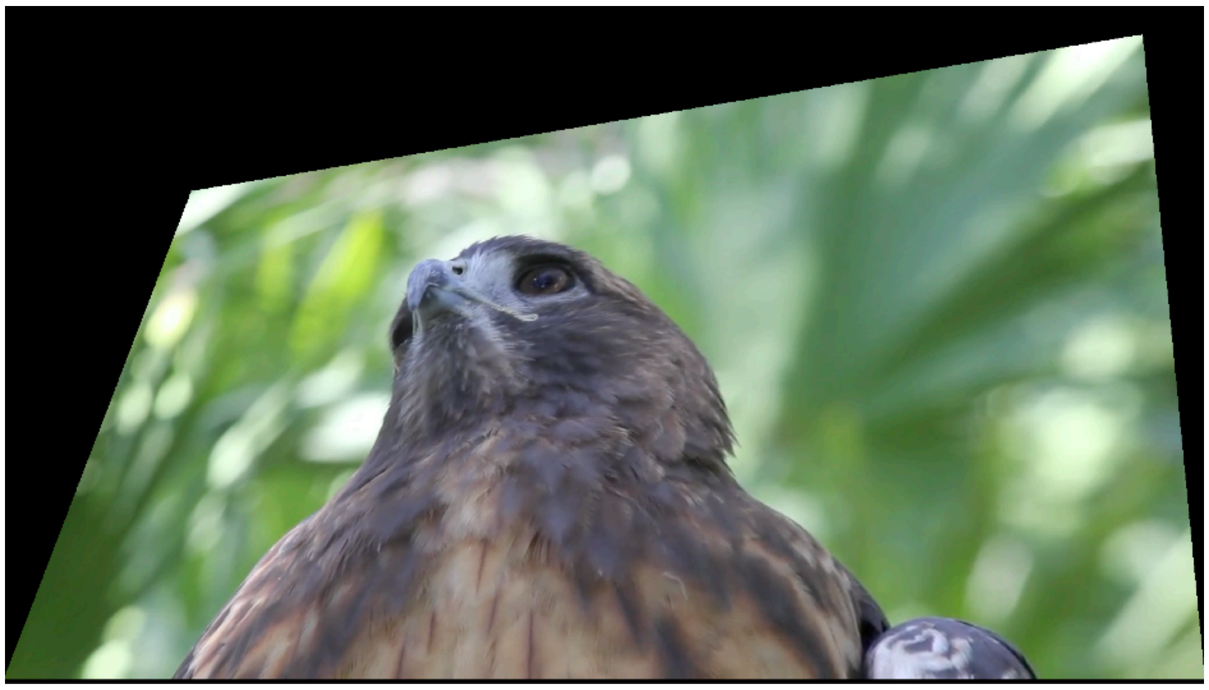
Changes all colors of the block by rotating them around the color wheel based on the value of a property. The value must be in the range 0…1, with the value 0.5 corresponding to 180 degrees of hue change causing red to becomes cyan, for example.

Saturate

Changes the color saturation of the block, either by a *Fixed Value* or based on the value of a property, with normal saturation corresponding to a value of 1.

Perspective

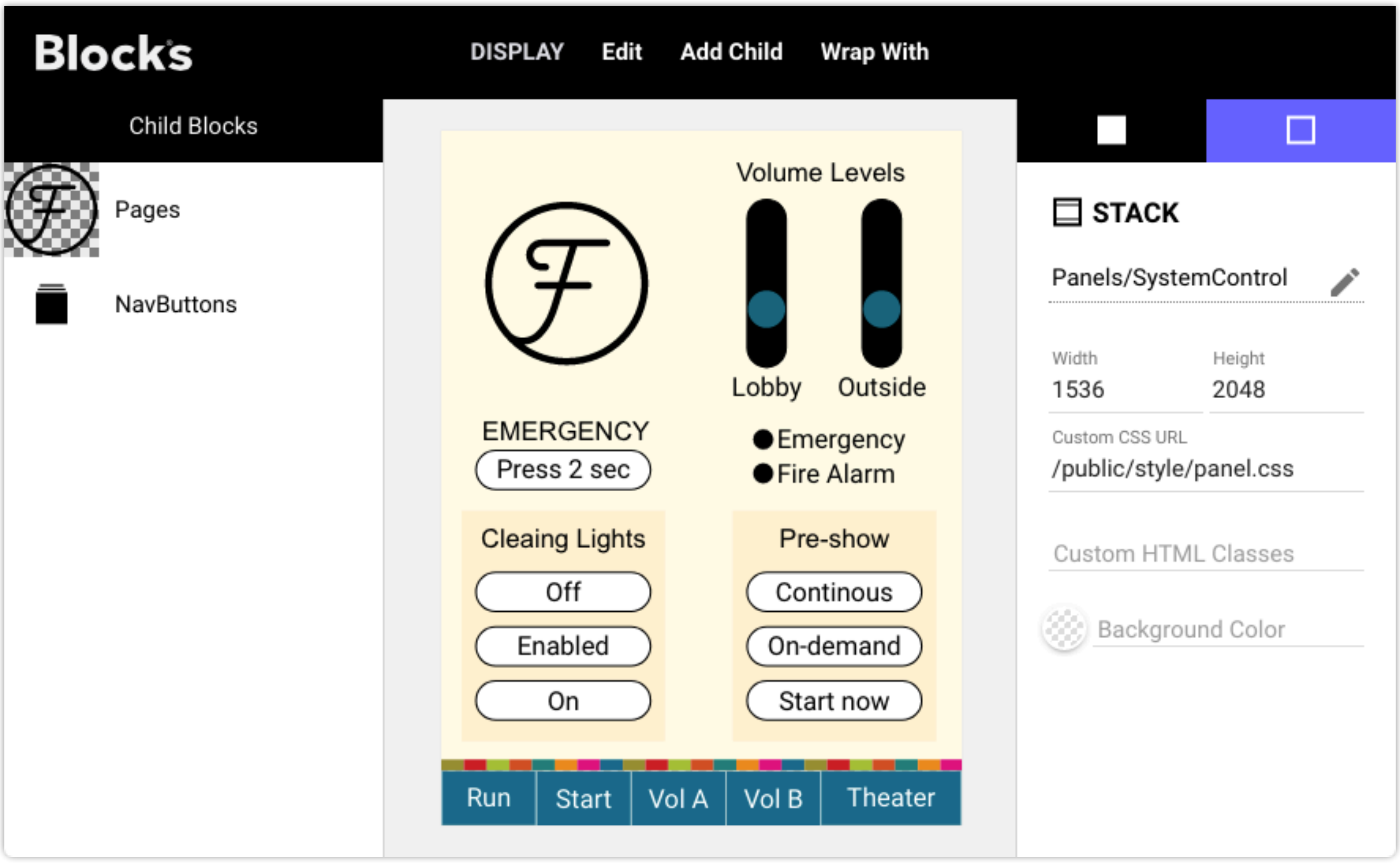
Moves the four corners of the block applied to by the specified amount, resulting in a perspective transform suitable for correcting for off-axis projection and similar. Like other behaviors, this applies to the Spot showing the block only – the effect is not visible in the editor.



A video with a Perspective behavior applied, moving the top left and top right corners.

8. CONTROLS AND PANELS

Use a Composition block to create a panel with buttons and other controls, as well as any other block types you may want to include. This can be used to build interactive kiosks for visitors, as well as sophisticated control panels for internal staff use (e.g., to control media playback on displays, power, lights, volume levels, etc).



Panels/SystemControl

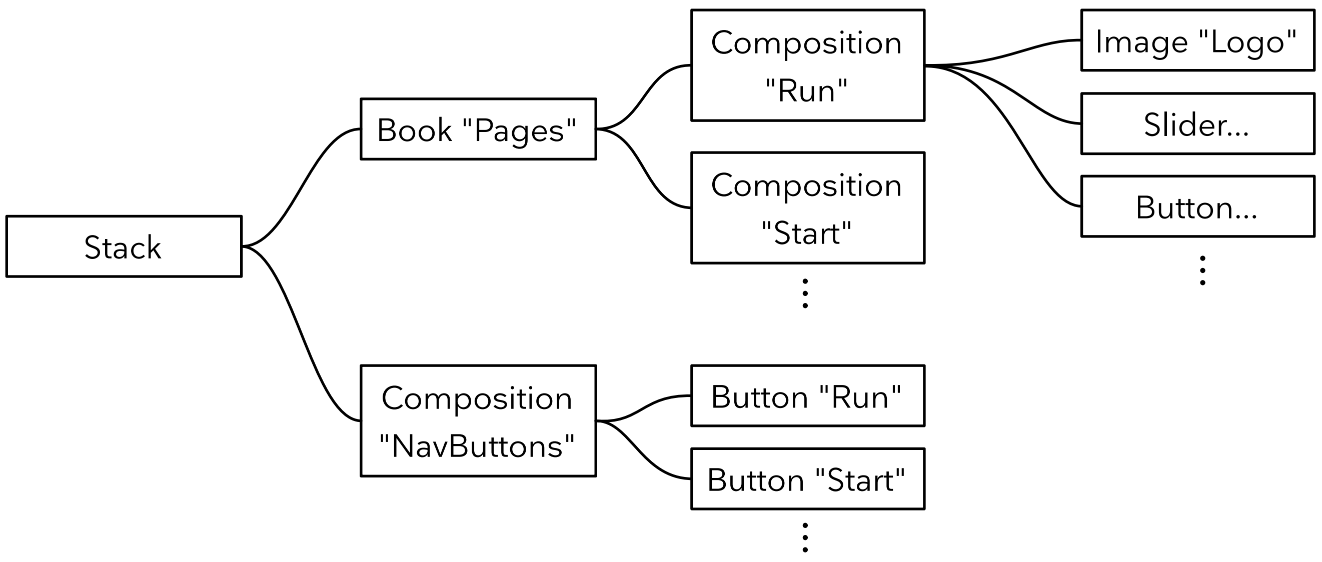
Width	Height
1536	2048

Custom CSS URL: /public/style/panel.css

Custom HTML Classes

A panel with buttons and sliders. The blue buttons in the navigation bar at the bottom have custom styling.

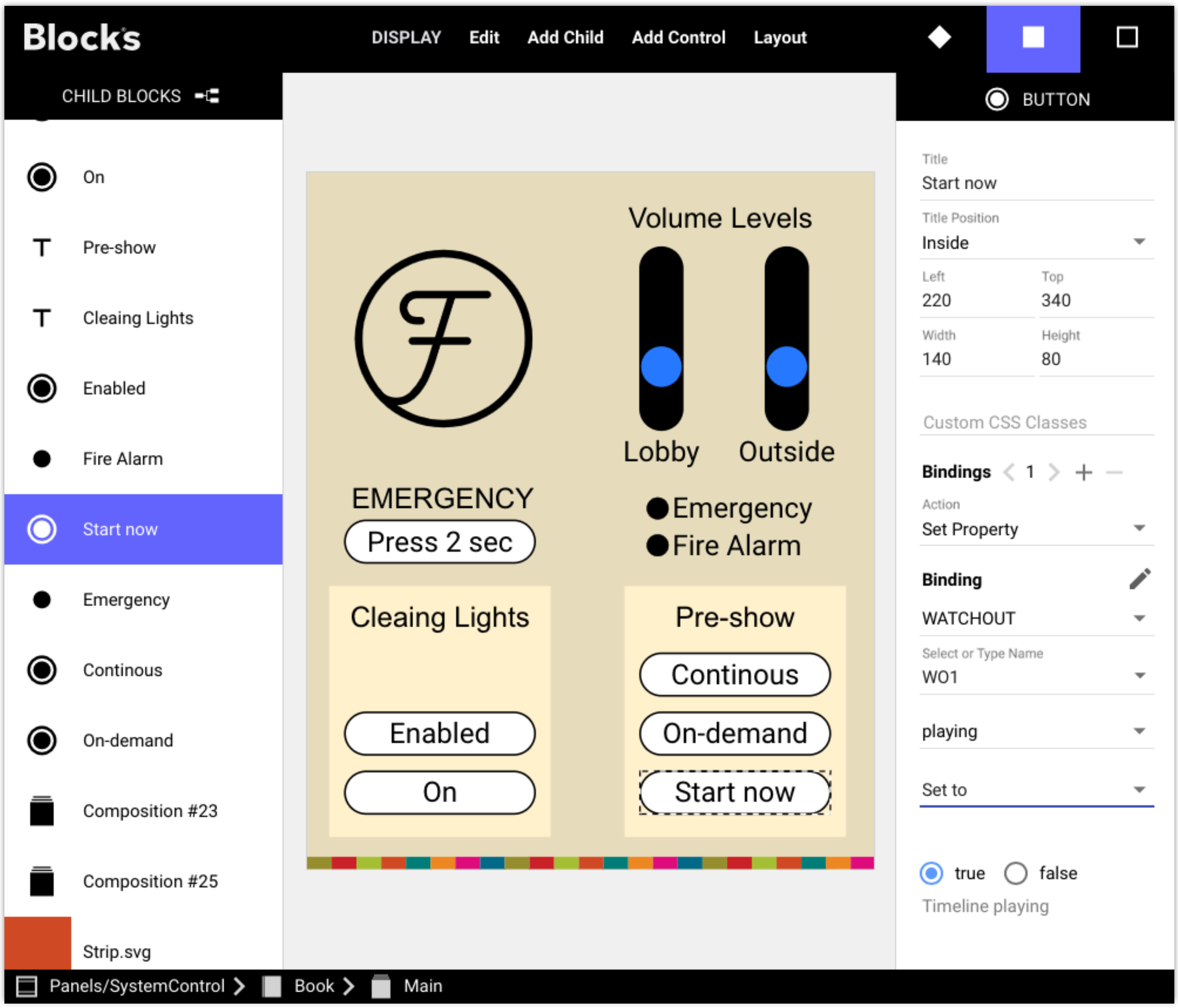
Buttons and other controls are added to a Composition, using the "Add Control" menu. The example shown above has the block structure shown below.



A control panel based on a Stack block, with a set of navigation buttons set to "Stick to Edge" at the bottom.

Button ◉

Use a Button to control some state of your system, to navigate within an interactive design, or to assign tags interactively to a spot.



Title: Start now
Title Position: Inside
Left: 220
Top: 340
Width: 140
Height: 80
Custom CSS Classes
Bindings 1
Action: Set Property
Binding: WATCHOUT
Select or Type Name: WO1
playing
Set to
[✓ true] false
Timeline playing

A button set to start a WATCHOUT show.

Name and Show Name

Specifies the name of the button, as well as where it should appear on or outside the button (or not at all).

Position

Allows you to numerically specify the position and size of the button. Alternatively, drag the button or one of its edges with the mouse. Turn on "Snap to Grid" under the outer block's settings to make it easier to line things up. You can also use the commands on the Layout menu to lay out multiple buttons or other controls horizontally or vertically.

Bindings

Initially, a button has a single binding. To add more bindings, click the + symbol. Use the arrows to navigate between bindings, as indicated by the digit between the arrows. The highlighting of the button is governed by its first binding.

Action
Selects the desired type of action. This may be to set a property (such as "play a video on a spot" or "turn on the lights") or a local action – one affecting the local spot only, such as go to another page or set a tag.

Set Property

After choosing "Set Property" as the button's action, you bind the button to a target property using the fields under Binding. In the example shown above, the selected button is bound to the "playing" state of the WATCHOUT subsystem named "WO1".

◆ **HINT:** As an alternative to using the menus under Binding, click the pen symbol next to the *Binding* heading and type, copy or paste the binding as text. This is particularly useful when setting up a large number of similar bindings.

The options shown under the binding target vary with the object and type of property selected. In the example above, the "playing" property is a true/false type (a "boolean"), hence the values indicated at the bottom. The last selector contains the following options:

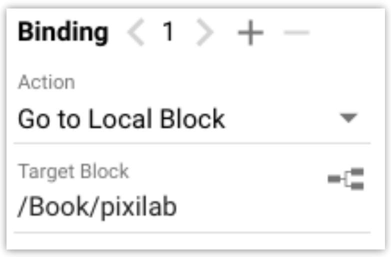
- **Set to Constant** sets the state of the property to the specified literal value when the button is pressed.
- **Set wile pressed** sets the state to one value while pressed, and to another value when released. This results in a momentary action, that's reset to the alternative value as soon as you release the button.
- **Toggle** flips the state between two possible values every time the button is pressed. This can be used to make a single button turn something ON or OFF, depending on the current state. Only available for the primary binding.
- **Append** appends the specified text string to what's already set. Only available for text properties.
- **Increase by** and **Decrease by** adds (or subtracts) a number to the property. Used to increase/decrease a numeric property, such as a light level, by a fixed value every time the button is pressed.
- **Set to Property Value** sets the state of the bound property to the current value of another property (the "Property Value Source"). Allows the value to be parameterized by a spot parameter or other suitable property.

Property Types
The values that can be set by a button vary depending on the type of the property selected. The following types are available:

- **Boolean.** Only *true* and *false* are available. This is the case with the "playing" property shown in the illustration above.
- **Number.** A numeric value, that in some cases may be negative and may contain a fractional part. Examples include the Volume level of a Display spot or an "Input" value for a WATCHOUT subsystem.
- **String.** A textual value, such as the name of a block assigned to a spot.
- **Time.** A time position, expressed in HH:MM.SS.fff, with the default unit being seconds. An example is the time position of a WATCHOUT timeline.

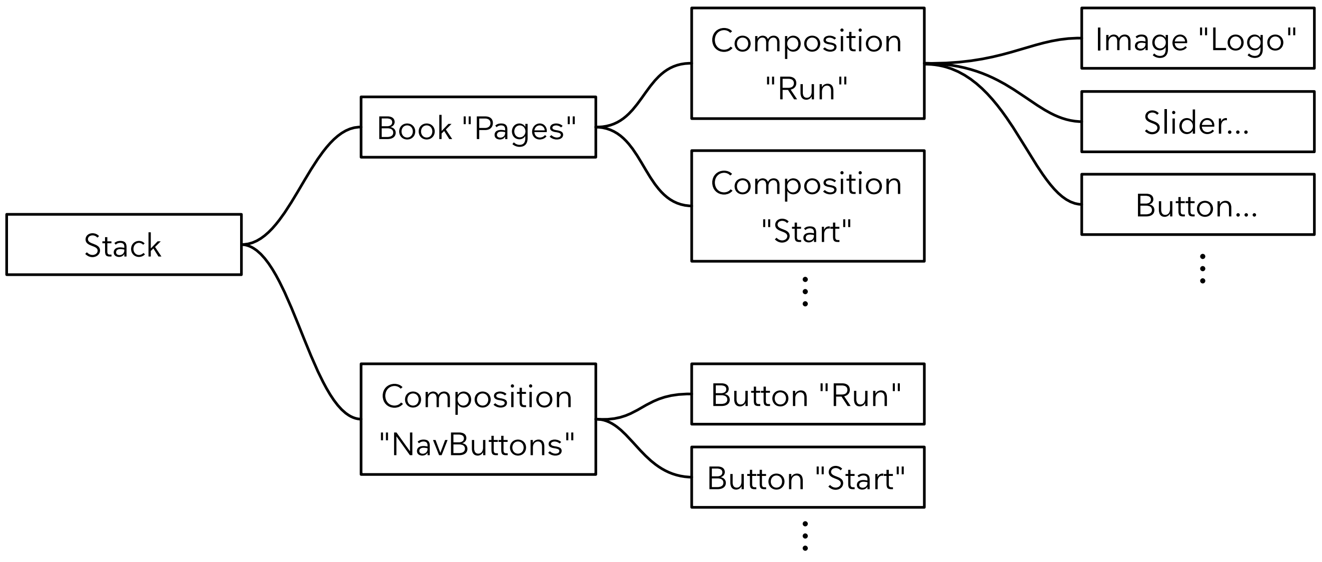
See the chapter titled "Properties and Functions" for a detailed description of the properties available for buttons.

Go to Local Block


Binding 1
Action
Go to Local Block
Target Block
/Book/pixilab

Navigate to another child block in the current root block. Pick the target block using the hierarchy selection button, then double-click the desired target block in the hierarchy view.

Alternatively, specify the target block using a text string. This states the path to the target page, either beginning from the page containing the button (a relative path) or from the topmost block (absolute path). Use the Copy Path button in the breadcrumbs area while editing a nested block to get its full (absolute) path inside the root block.



For example, if you want to make the “Start” button shown in the illustration above reveal the “Start” page in the Book, you can specify the path in either of the following ways:

- **As “/Pages/Start”.** This specifies an *absolute path* starting from the top level Stack block which contains a Book named “Pages”, which in its turn has a Composition named “Start” .
- **As “../Pages/Start”.** This is a relative path, beginning at the page containing the button. The button is on the “NavButtons” Composition in the top level Stack block. The two periods (..) mean “go up one level”, which gets you to the top level Stack block, from which you then continue down into “Pages” and to “Start”.

In this case, the absolute form is preferred, as it is both shorter and easer to understand. However, in more complex and deeper block hierarchies, it is often easier to use relative paths.

◆ An *absolute path* begins with a forward slash, while a relative path doesn’t. The path syntax is similar to how file system paths are specified on Unix-based operating systems, such as macOS.

Locating a Target Numerically or Incrementally
As an alternative to named targets, you can use the numeric position of the target. Simply a number instead of the name, where the first child block has number 0 (zero). In the example above, you can go to the second page in the “Pages” book by specifying the path “/Pages/1” (where 1 is the second page, since the first page is numbered 0).

◆ A number in the path always specifies a target index. Thus, don’t name a page with just digits if you want to go to that page by name. If you want to use a number here, prefix it with some text, such as “page2”, “page3”, etc.

A similar syntax can be used to move incrementally forward or backward by a specified number of pages. To move two pages forward in the book, specify “/Pages/+2”, or one page backwards as “/Pages/-1”. This syntax can be used to jump forward or backward, but will not wrap around once reaching the first or last page/slide.

Finally, using just a + or - (without any following digit) will move by one step, wrapping around in a slideshow or book from the last page to the first, or vice versa. Moving forward through a slideshow in this way will apply any transitions.

Locating Targets from Inside a Reference Block
When using a Reference block to bring in another top level block, you can navigate within the referenced block using any of the methods described above. However, if you need to target a block *outside* the referenced block, you need to use a slightly different syntax. This is due to the fact that the absolute target syntax (beginning with a single forward slash, as discussed above) operates based on the root of the referenced block – just as if the referenced block was used on its own.

To break out of the referenced block, up to the ultimate root of the outermost block in the reference chain, use an absolute syntax that begins with *two* forward slashes //. You can then navigate downwards from there into regular child blocks or reference blocks (which may in their turn reference other blocks in multiple levels).

◆ This double slash syntax is not required when locating targets using a task statement, as such references *always* start at the ultimate root level.

Exclude from “Go Back”
Normally, block navigation initiated by buttons is recorded in the navigation history, allowing you to use a “Go Back” button to backtrack through this block sequence. However, in some cases – such as when using buttons to flip pages in a book – this may not be desired. If so, select this option to make any “Go Back” button skip those intervening navigation steps.

Locate Spot

This action is used together with a Locator, as an alternative to other means of location. It has no effect unless there’s an active Locator. Such buttons can be used as a menu-based way of specifying the location, rather than typing an ID or scanning a QR code. If you're not using any of the other means of location, the Locator’s keypad will not be shown. But it must remain active for the “Locate Spot" action to work. Leave the target spot name field empty to close any currently open Locator.

- ◆ When using this method, enter the *name* of the spot into the “Spot Name” field, not its Location ID. A Location ID is not needed for this method to work. If the target spot is in a Spot Group, type the full path beginning at the outermost group name, with each level separated by a period.
- ◆ **HINT:** You may leave the target spot field empty to close any current Locator, rather than navigating to a new spot.

Close Locator

Closes any currently open Locator. This is the same function as the Locator’s built-in “Back Button”, but provides more control over its look and positioning.

Go Back

This action is equivalent to pressing the browser’s *back* button. The browser’s URL bar and navigation buttons are often hidden when used with Blocks. This action can then be used to navigate back to a previous block. It’s often used in conjunction with the “Go to Local Block” action (described above), as a way to backtrack such local navigation.

Select/Reset/Toggle Local Tags

The tag-related options are used to add/remove tags. A tag is a single word, or a few words, used to affect the behavior of the spot. Specify multiple tags by separating them with a comma. The following options are available:

- **Toggle Tags.** Pressing the button will add tag(s) if not already set, or remove set tag(s).
- **Reset Tags.** Pressing the button will remove the specified tag(s).
- **Select Tags.** Sets the specified tag(s) out from a set of tags. That is, the tags specified under “Tags” will be *set*, while any additional tags specified under “Of Tag Set” will be *cleared*. For instance, to select the single language tag *swe* out of the set *swe, eng, ger*, enter the former under Tags and specify *swe, eng, ger* under “Of Tag Set”.

Tags can be assigned to a spot from the outset using the “Tags” field in the spot’s dialog box. Additional tags can then be applied using buttons, as described above, or using tasks and scripts. A tag set is local to the spot, and has no effect on other spots.

You then use the current set of tags in conjunction with a Tag Selector or Synchronizer block to choose which child block to display. This is useful for language selection, and other similar, local customizations.

Setting a Spot Parameter
Chose “Set Spot Parameter”, then chose or type the name of the parameter and its desired value. See “Parametrization of Property Paths” for more details.

Button Feedback

A button will light up to indicate the state of the first (or only) property or local behavior it controls. Specifically, it will light up under the following conditions:

- When controlling a System property, the button will light up when the property has the "Set to" value, or the first of the two specified values when "Set while pressed" or "Toggle" is selected.
- When set to Local; Go to Page, it will light up when the specified page is visible. This is particularly useful for "navigation bar" buttons, as in the example shown in the illustration above, where you can see the button as well as the target page at the same time.
- When set to Local; Select/Toggle Tags, it will light up when all tags specified are set.
- When set to Local; Set Spot Parameter, it will light up when the parameter has the specified value.

  ◆ This status indication isn't shown in the Blocks editor; only when interacting with the real Display/Visitor spot.

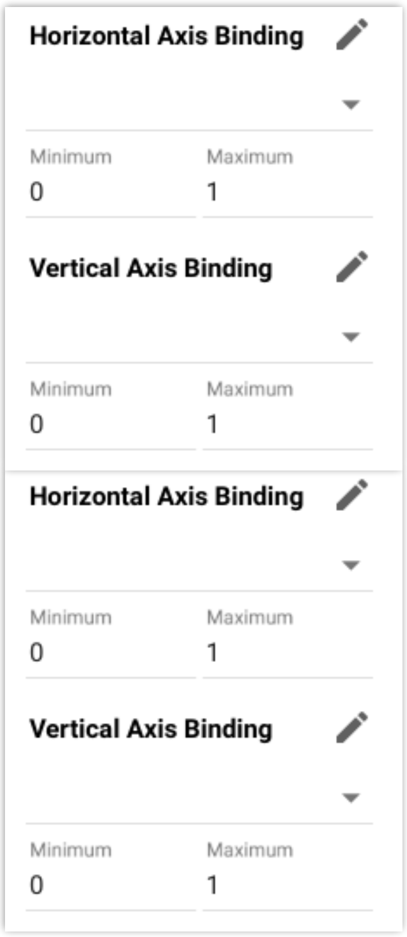
Indicator ●

An Indicator is similar to a Button, as described above, but provides only its feedback functionality. It can only be bound to a single property.

Slider

A Slider controls the value of a numeric property gradually. It's suitable for properties such as volume or brightness. You can specify the maximum and minimum values that can be set using the slider (which may be smaller than the maximum range supported by the property). It can be used with boolean (on/off) properties as well, in which case the slider just acts as a switch.

Connect the slider to the target property in the same way as described above under Button, System Behaviors. A slider also indicates the value of the property it controls. The **Spring Back to Center** option sets the controlled property values so that the slider returns to its center position when it is released.



Horizontal Axis Binding

Minimum	Maximum
0	1

Vertical Axis Binding

Minimum	Maximum
0	1

Horizontal Axis Binding

Minimum	Maximum
0	1

Vertical Axis Binding

Minimum	Maximum
0	1

Bar

A Bar behaves like a Slider, but can not be used to control the property; only to indicate its value.

Joystick

A two-dimensional slider, with a knob that can be dragged horizontally as well as vertically. This control type is particularly useful for pan/tilt control of a moving head light or a PTZ-style camera. It must be bound to two numeric value properties – one for each axis. For each of these properties, you can specify the allowable range.

The **Spring Back to Center** option sets the controlled property values so that the joystick returns to its center position when it is released.

◆ **HINT:** You can move the bound property between any two extremes. For instance, by setting the Maximum value less than the Minimum value, the value will go backwards when moving the joystick right/down.

Text T

As part of a control panel, a Text block can be used to add labels and similar static textual information. It can also be used to display the value of system properties such as numeric values, time positions, string variables or boolean (on/off) data. See "Text" in the Blocks chapter for more details.

Custom Styling

While text (as well as buttons and other controls) uses a simple font and style by default, you can override this using cascading style sheets (CSS), as described here: https://pixilab.se/docs/blocks/custom_styling.

Text Input

Use a Text Input to enter numeric values or text. Bind the Text Input to the desired property, such as a realm variable. When bound to a numeric property, only numbers can be entered.

- **Accept Text Entry** specifies whether to accept text automatically as you type or only when you explicitly press Enter or leave the text field (e.g., by pressing the tab key).
- **Multiple Lines** allows multiple lines of text to be entered. When selected, pressing Enter/Return while entering a value makes a new line, rather than accepting the entered value. Resize the control vertically to view multiple lines of text.
- **Autofocus** activates the text input field automatically as it appear son screen, without having to click it.

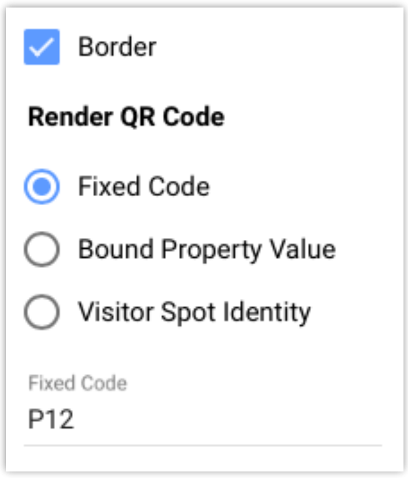
QR Code

Displays a QR Code on screen, with either fixed or programmable data.

◆ **NOTE:** This is a premium block type, requiring additional license options in order to be displayed.

Border automatically applies a white border around the QR code for improved readability against a colored background.

The data represented by the QR code may be a number or a short text, but is often a URL. It may originate from one of the following sources:



✓ Border

Render QR Code

● Fixed Code

○ Bound Property Value

○ Visitor Spot Identity

Fixed Code
P12

- A **fixed code**. Just type what you want to be rendered as a QR Code.
- The value of a **bound property**. This may be a Spot-local parameter, a Realm variable, or any other property in your system.
- A visitor's mobile **device identity**. This option allows a visitor to be identified by presenting a QR code shown on her mobile device's display. This can then be read by a QR Scanner block at a kiosk being visited, thereby letting Blocks know who's where. This option is particularly useful when using the Visitor Data Collection feature of Blocks to follow your visitors for a more personalized experience.

If you intend to use the QR Code together with a Locator, the QR code should contain one of the following:

- A plain Location ID number. This results in the shortest QR code, increasing its readability. It's also useful in cases where you want to present both a location ID number and a QR code to your visitors.
- A Spot name or path (with path segments separated by a period). This operates independently of any Location ID, and not such ID need to be assigned to the Spot.
- A URL, indicating which Spot to locate using a *spot* query parameter appended to the URL, with a value set to the Location ID or Spot path, as described in the previous two points.

The last option allows you to use a QR code from a mobile device to do both of the following:

1. Open a Visitor Spot web page (using the URL part).

2. Locate the correct spot in Blocks (using the *spot* query parameter).

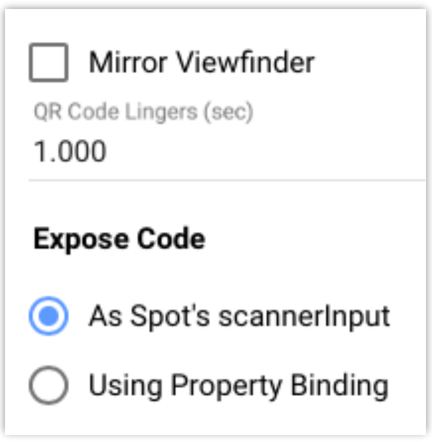
Most mobile phones have an inbuilt QR code scanner – for instance the iOS Camera app, Google Lens on Android or using a separate QR scanner app – taking care of the first step. Here's an example of such a QR code URL (here using the "Spot path" option, rather than an ID number):

```
https://pixi.guide?spot=Entrance.Welcome
```

QR Scanner

Uses a camera, such as a webcam connected to a Display Spot running PIXILAB Player software, to scan a QR code, presenting the result as a property. While this can be used to scan any QR code in a generic way, it's particularly useful in conjunction with a QR Code block presenting the identity of a visitor, as described in the previous section.

◆ **NOTE:** This is a premium block type, requiring additional license options in order to be displayed. When presented on other types of Spots than those running PIXILAB Player, your Blocks server must support https and the Spot must connect using https.


☐ Mirror Viewfinder

QR Code Lingers (sec)
1.000

Expose Code

[✓ As Spot's scannerInput]
Using Property Binding

- Select **Mirror Viewfinder** to mirror the preview. This sometimes make the positioning of the image more natural. It has no effect on the scanned code.

- When scanning a code, the property exposing the result will be set to the scanned code for the duration specified by **QR Code Lingers**, and then reset to empty string. This time needs to be long enough for the code to be recognized by the script or task triggered by scanning the code, yet short enough to not get in the way of any subsequent scanning of the same code. In general, one second is OK here, but a shorter time may be needed if you must be able to scan the same code repeatedly with higher frequency.

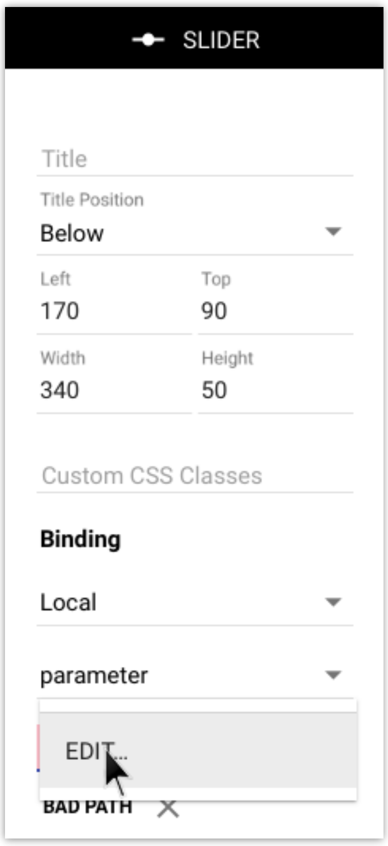
- Chose to **expose** the scanned code to the Spot's *scannerInput* property or to an arbitrary property (such as a Realm variable or a Script property).

9. SPOT PARAMETERS

You can add arbitrary custom properties to Display spots. These are called “spot parameters”. They can be used for a variety of purposes, such as:

- Local or system-level interactivity, sometimes in conjunction with behaviors.
- Exposing custom properties on a Spot, making those accessible from other parts of Blocks.
- As parameters in property bindings, making those more flexible.

Spot parameters can be defined for each top level block (as *formal parameters*, making those accessible in this block). They can then be overridden on a per-Spot basis, thereby defining a spot-specific initial value. Furthermore, parameter values can be changed at runtime, as they are exposed as properties on the Spot. Spot parameters, like other system properties, have a specific *type*, such as number, boolean (true/false) or a text string.



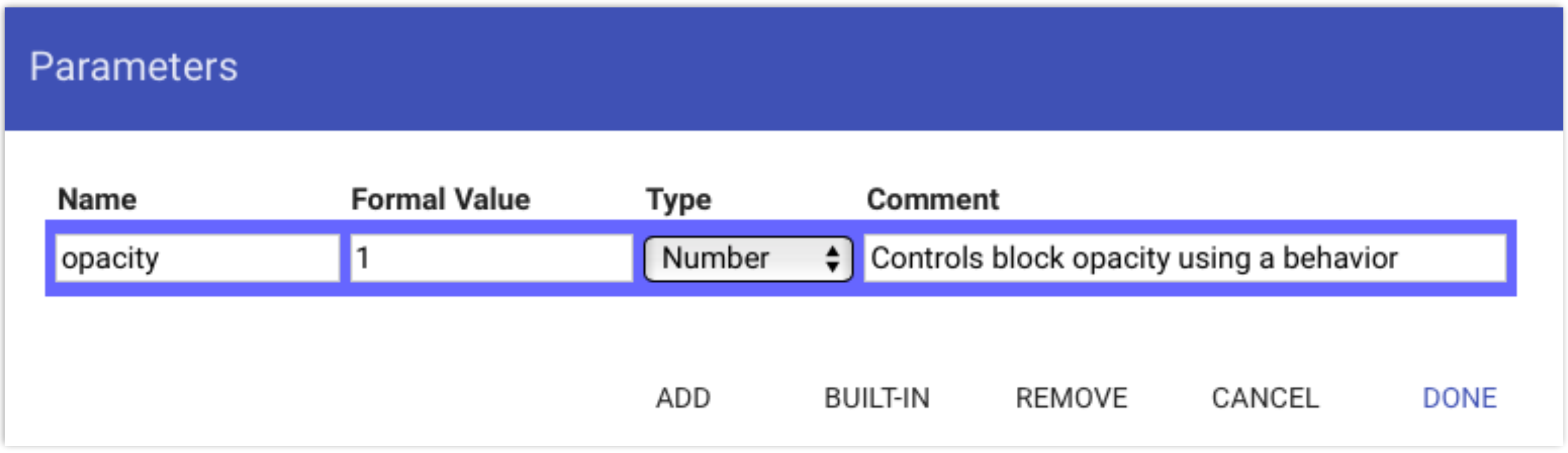
Title Position: Below
Left: 170 Top: 90
Width: 340 Height: 50
Binding
Local
parameter
EDIT...

Defining Parameters in a Block

As parameters are often used by a block, they need to be known to that block. Initially, there are no parameters defined. Either use the “Edit Parameters” command on the Edit menu. Or, to establish and use a parameter as part of a property binding, do as follows:

1. Choose Local under binding, as shown in the illustration to the right.
2. Choose parameter on the next dropdown.
3. Choose EDIT… on the third dropdown

Any already existing parameters appear on the third dropdown, allowing you to bind to that parameter as to any other property. If there aren’t yet any parameters defined, this dropdown contains only the EDIT command.



Name	Formal Value	Type	Comment
opacity	1	Number	Controls block opacity using a behavior

1. Click ADD to add a new parameter.
2. Enter its name, which must begin with an alphabetical character and contain only alphabetical characters, digits or underscore.
3. Enter its formal value. This is not required if the parameter is of type String.
4. Select the parameter's type.
5. Enter a comment that describes what you intend to use this parameter for.
6. Click SELECT to bind to this new parameter.

After defining a parameter, it can be used anywhere within this block. As this is a numeric value, initially set to 1, it could, for example, be used to control the opacity of some other block using a Behavior bound to this parameter.

Formal Value

A parameter's formal value serves the following purposes:

- Provides a value for parameter while editing the block in which it is defined.
- Provides a default value if not overridden by a Spot or set programmatically.

The first point is particularly important when using parameters in property paths, as the formal value is then used to resolve the property path, finding a representative system property to bind to. Hence, substituting the bracketed parameter with formal value must result an existing property path. If not, change the formal value of the parameter to one that makes the path valid.

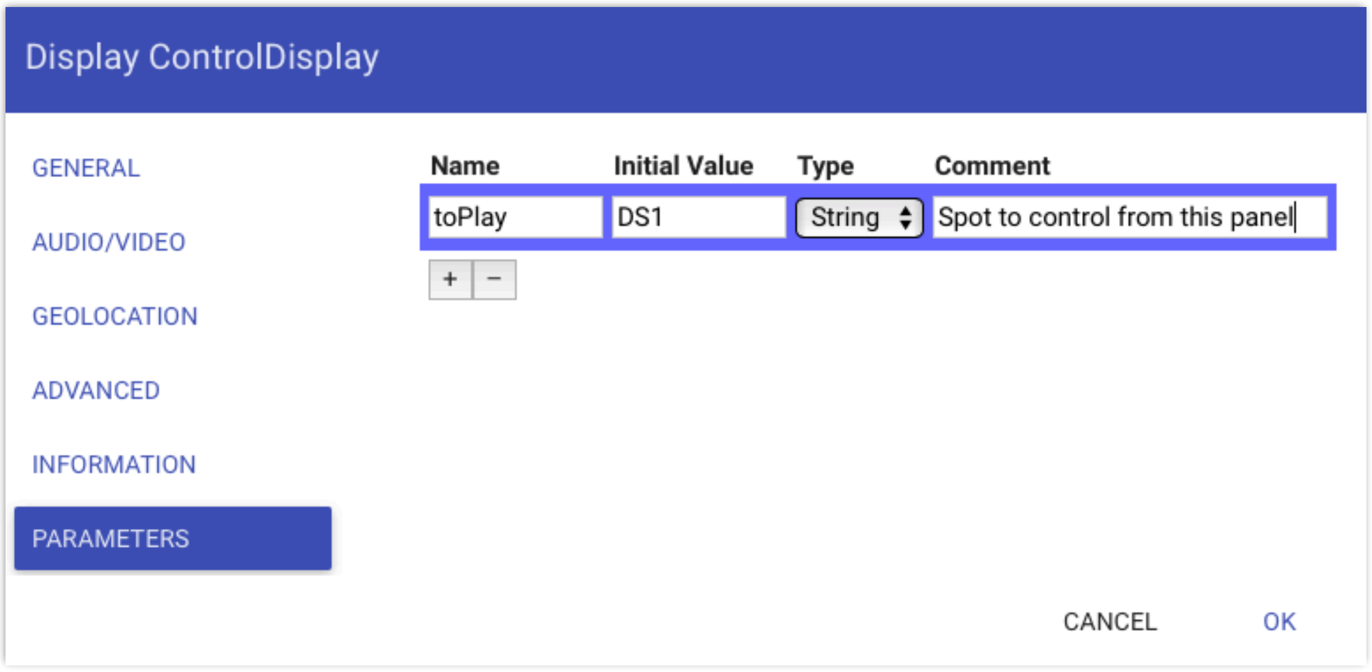
Built-in Parameters

In addition to parameters you define yourself, there are a few built-in parameters, where Blocks will automatically assign the proper value at runtime. To uses such a parameter, you must add it using the BUILT-IN button next to the ADD button.

- **ThisSpotPath** corresponds to the full path to the Spot used to play the block. This lets you refer to "the current spot" in property paths. Note that you must manually enter the path to a representative spot as the formal value. This is required for editing purposes, although the actual value will be set automatically at runtime.
- **LocatedSpotPath** corresponds to the Spot currently located using a Locator, letting you target that spot using buttons and other controls. The value of this parameter will be set automatically at runtime by the Locator, and will be undefined unless there's currently a located Spot.

Defining or Overriding Parameters for a Specific Spot

One way to use spot parameters is to override it for a specific Spot. By entering the same parameter name into the list of Parameters in the Spot's settings, you can override its initial value for that spot only.



Name	Initial Value	Type	Comment
toPlay	DS1	String	Spot to control from this panel

This makes it possible to use the same block across multiple spots, with customized functionality (either in terms of behaviors or property bindings) based on the spot's local parameter value. For more on this use case, see property path parametrization below.

Accessing Parameters at Runtime

You can change the value of a parameter at runtime in either of the following two ways:

- Using a control – such as button, slider or text input – bound to the parameter using the Local.parameter.*paramName* path, as shown in the illustration at the beginning of this chapter. Moving the slider changes the parameter's value in the local spot only.
- Setting it through the spot to which the block is assigned. This can be done from anywhere in the system, including controls and tasks. It uses a property path such as Spot.*spotname*.parameter.*paramName*.
- When using a Visitor Spot with Visitor Data Collection enabled, fields in the visitor's data record decorated with @spotPara-meter will reflect the value of the spot parameter, and vice versa. This is an advanced feature, accessible only from User Scripts.

The second method is available only when the spot named *spotname* is online and has a parameter named *paramname*.

Determining the Value of a Parameter

The current value of a spot parameter can come from a number of different sources. Those value sources are used in the follow-ing order, with the highest priority being number 1.

1. Value set at runtime using a Local.parameter.*paramName* or Spot.*spotname*.parameter.*paramName* property path.
2. Values defined through a query parameter in the Spot's URL.
3. The *Initial value* set in the spot's settings. This applies also to Visitor Spots.
4. Formal value specified by the block currently presented by the spot.

Interactivity with Behaviors

Spot parameters can be used to drive interactivity in conjunction with behaviors. As an example, you could make a page that shows three different videos, side by side. A button under each video can make its own video play, while pausing any other of those three videos.

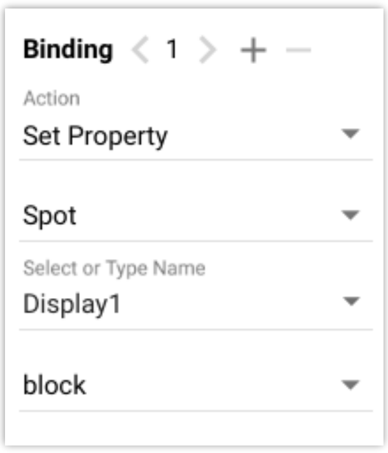
1. Add three videos to a composition.
2. Add three buttons, one under each video.
3. Connect each of those buttons to a local parameter named *videoToPlay*, of type String with no formal value, with the button setting the parameter to the name of the corresponding video (or just the names A, B and C for the sake of simplicity).
4. Apply a Play behavior to each of the videos, binding to the same local parameter named *videoToPlay*, and matching agains the same three names.
5. Assign this block to a spot.

Now, pressing any of those three buttons on that spot will trigger its corresponding Play behavior, making its video play while pausing any of the other videos.

Custom Spot Properties

Parameters established by a Block are also exposed as properties on Spots with that block assigned. Thus, you can trigger associ-ated actions from outside as described above under "Accessing Parameters at Runtime". This allows you to define your own spot properties using parameters, which in their turn can control or influence behaviors and property bindings.

Use in Property Paths



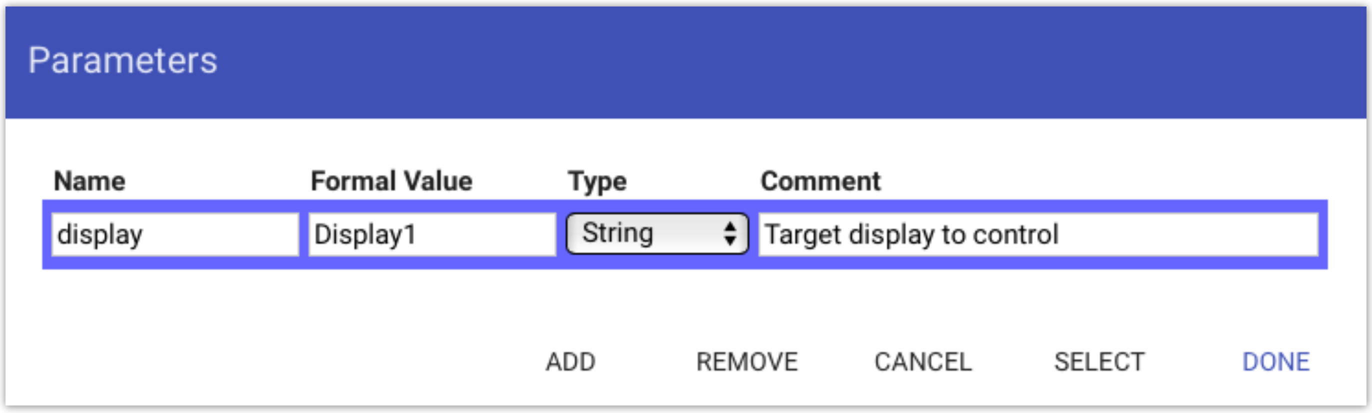
Set Property
Spot
Select or Type Name
Display1
block

The path to a target property is selected using a set of menus, as shown in the illustration to the right. This results in a fixed property path, meaning that this button is always bound to the same property (namely “Spot.Display1.block” in this example) regardless of its context.

In some cases, though, you want the target property to vary with the context. For instance, consider a set of Display spots along a wall, each one controlled by its own iPad on a pedestal. Those displays show the same content (block). The iPads all show the same control panel used to manage the display behind each iPad. However, you want each iPad to control *its own* display spot. This can not be solved using buttons with fixed property paths, since part of the property path (in this example, the Spot name) would need to vary for each iPad/display pair.

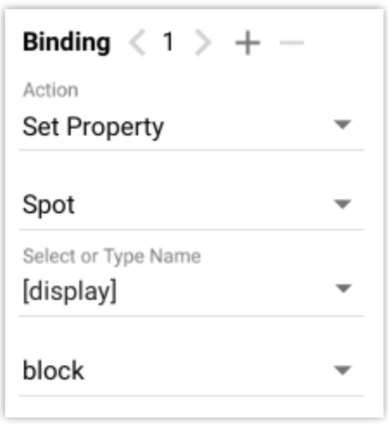
One solution could be to duplicate the block shown on the iPad, making one copy corresponding to each target display. The paths can then be modified to match the name of the desired target display. However, while this works, it results in a solution that’s hard to maintain and update with new functionality for the iPads, as there are now three different blocks to maintain.

A better solution is to use a *parameter* instead of the hard-coded display name. Whenever this is possible, an extra item appears at the bottom of the drop-down menu, titled [PARAMETER]. Selecting this item displays a dialog box allowing you to define parameters along with their *formal values*.


Parameters

Name	Formal Value	Type	Comment
display	Display1	String	Target display to control

Add a parameter named “display”, as shown above. Set its *formal value* to something that can be used to test the functionality. This formal value is used by the editor to obtain the set of valid child properties, so it must be a name of a representative display spot in this example. Also, this value will be used as a default value for the parameter if no spot-specific parameter with the same name is found (see “Defining Spot Parameters” above).



Set Property
Spot
Select or Type Name
[display]
block

To use this newly added parameter in place of the hard-coded display spot name, select it as shown above, then click the SELECT button. This puts [display] into the drop-down menu instead of the fixed display name. Since this parameter resolves to the same name (here “Display1”), everything still works the same.

However, when you deploy the block containing this button to an iPad spot, the value of this parameter can be overridden for each iPad using a similar list of parameters found in the spot’s settings (see “Defining Spot Parameters” above). By overriding each iPad’s *display* parameter with the desired target name, each one will control its own display spot even though they all show the same control panel.

The same method can be used to parametrize any type of property binding, including network devices, WATCHOUT clusters, relays, etc. Keep in mind that the value assigned to the parameter on the spot takes precedence over the formal value specified for the parameter with that same name in the block. You need to make sure that the resulting value of the parameter makes sense for your system, and update it accordingly if you rename objects. If there’s no matching object found, the button will have no effect, and an error message indicating this problem will be logged.

10. PIXILAB PLAYER

PIXILAB Player is free software you can install on a NUC or similar computer for use as a Display Spot. In addition to basic bowser functionality, it has advanced functions such as:

- Easy configuration of audio and multiple displays, including display resolution refresh rate and rotation.
- The ability to remotely power down and up the Display Spot as well as any attached display.
- Boots either from a USB memory stick or internal SSD.
- Supports general content caching for reducing the data traffic as well as limited offline playback even when not connected to the Blocks server (separate licenses required).

With *Offline Use* enabled, the default block and all its media is stored locally on the player.

11. VISITOR DATA COLLECTION

Blocks allows for *visitor data collection*, based on the location awareness provided by the Locator block, as well as other forms of data collection. This feature, which is licensed separately, adds the following capabilities:

- Identify individual visitors either based on their mobile device, a token of some kind (such as an RFID bracelet), or a combination of both.
- When using mobile devices, based on the Visitor Spot, you can interact with each individual visitor, providing a tailored experience with detailed control over what appears on that person's mobile device.
- A visitor can enter information or preferences –such as name, favorite color, age, etc – through her mobile device, using separate registration kiosks or other points of interaction.
- When visiting a particular location, this information can be collected either based on the visitor scanning a "token" or using a Locator block on a mobile device.
- Use a Camera Block to collect pictures taken by (or of) the visitor, showing such pictures later during the visitor's journey.
- Collect individual scores from gaming stations, or similar.
- Present visitor information, scores, pictures , remaining credits, or similar, when visiting relevant stations.
- Aggregate information from multiple visitors, presenting the result as "high scores" or other forms of collective data.

These capabilities are based on the following key features:

- An enhanced Visitor Spot, allowing you to programmatically interact with each individual visitor. This provides most of the scripting capabilities previously only available for Display Spots, such as the ability to present personalized content to each visitor.
- A built-in, high-performance database, managed by Blocks, collecting and logging all visitor interactions into individual visitor *data records*.
- User Script enhancements for creating, using and managing the lifecycle of visitor data records as well as interacting with individual Visitor Spots.

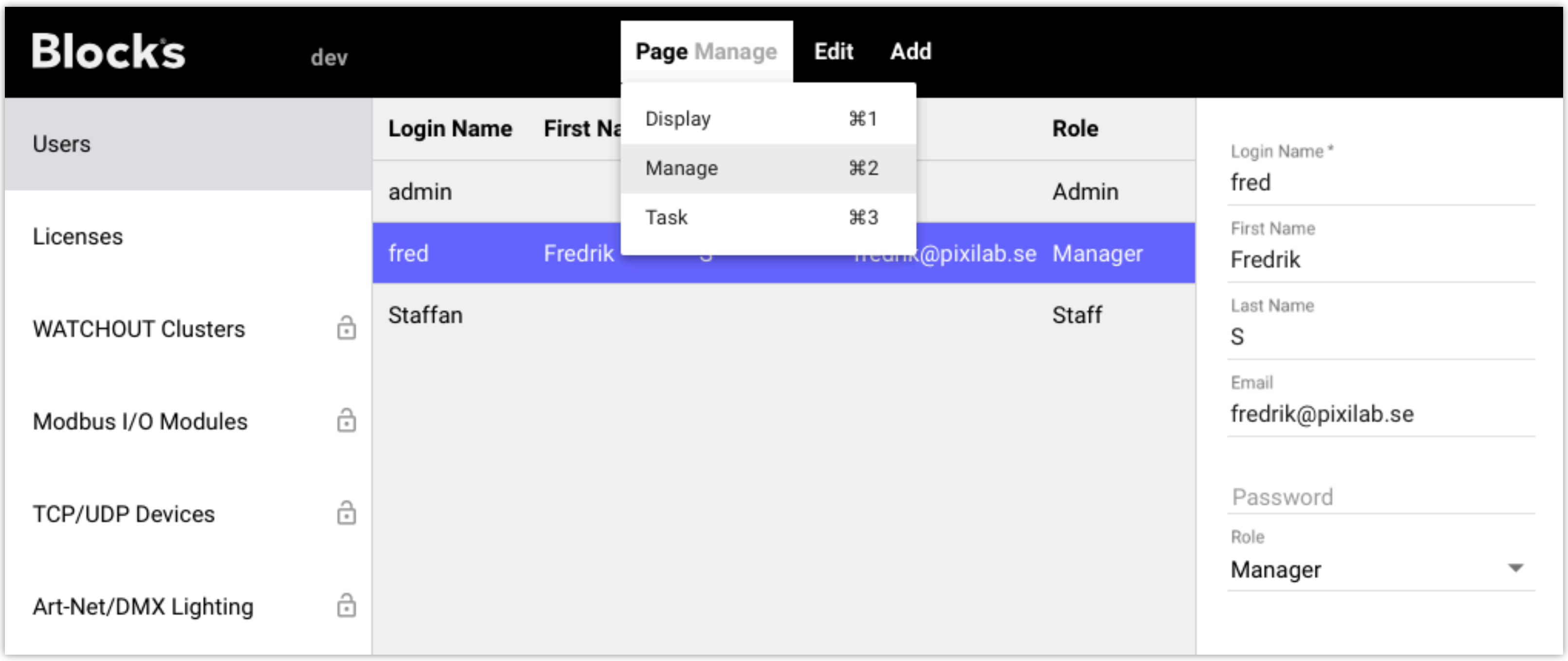
A User Script defines the database schema used to collect visitor information, and is responsible for creating the data record, updating and using its content during the visit, and finally discarding the record. This mechanism is specifically designed to manage data during a single visit, with data being discarded (or archived) as the visitor leaves or at the end of the day. It is not intended as a long-term data storage mechanism. (There'a separate Blocks API for dealing with external SQL databases, if you need that.)

Privacy Considerations

Since this mechanism potentially collects personally identifiable information, you need to take the regulations in your jurisdiction into consideration (e.g., the European GDPR). If such data is collected, you may need to inform your visitors about this and possibly obtain their permission to do so. The fact that data can be discarded when they leave (or at the end of the day), alleviates some of the potential difficulties involved (such as the "right to be forgotten").

12. MANAGE

The Manage page is used to add and configure a variety of objects, including users, WATCHOUT clusters, I/O and network devices, and more. You access this page using the Manage command in the menu bar. Then select which area to manage using the list on the left hand side.



Login Name	First Na	Role
admin		Admin
fred	Fredrik	Manager
Staffan		Staff

Login Name * fred
First Name Fredrik
Last Name S
Email fredrik@pixilab.se
Password
Role Manager

Users

The system contains at least one user, always called “admin”. This user can not be removed or renamed. You can (and should) change the password of the “admin” user. You may also set its first/last name and email address.

Additional users can be added through the “Add” menu. You must specify a “Login Name”, which is used to log in to the system, along with a password. You may specify additional information to further identify the user. The Role setting specifies the intended function of the user, with the “Admin” role having full access to all areas of Blocks (including adding users and setting their roles). See “Authorization and Roles” for more details on roles and permissions.

◆ If your Blocks system has been configured to use an external single-sign-on (SSO) provider, such as Active Directory, any users defined under Manage are ignored. If so, contact your system administrator for details.

Licenses

The content of your Blocks license is shown under “Licenses”. Licensing is managed by either of the following methods:

- Using a hardware license key connected to a USB port of the computer acting as the Blocks server. As an alternative, the USB license key may instead be connected to another computer, accessible over the network from your Blocks server.
- Using a cloud-based license. This is particularly suitable for running Blocks in an environment where a physical USB port isn’t available, such as a virtual private server (VPS) or other virtualized environment. This method requires that your Blocks server has internet access.

Licensable options

Spots are licensed separately, as are some premium block types. For each licensed feature, you can see:

- The number of Spots or other counted functions you have licensed.
- The number currently used or configured in the system.

- The number of Spots actively used at this time.
- For premium blocks, whether the block type is licensed. If so, you may use an unlimited number of such blocks.

There are two low cost licenses for special purposes, referred to as Producer and Evaluation, with further restrictions:

- Producer license. Blocks will shut down automatically after four hours, and must then be restarted manually.
- Evaluation license. The same limitation as the Producer license, plus an expiration date (typically 30 days from first use).

Offline Spots

The *Offline Spots* license item (previously known as "Block Caching") controls how many players you may enable Offline Spot usage for. This license is in addition to the Display Spot license, and applies to the Offline function only. Players for which you have enabled this feature are tracked in the Licenses-XXXX file, located in the model directory of your Blocks root, as such entitlements can not be revoked. Do not rename, delete or share this file across servers or Blocks roots.

General Content Caching

Video, images and other content can be cached on supporting players, thereby reducing network traffic for repeated playback. This is supported by PIXILAB Player version 6 and later when a suitable storage medium, such as an internal SSD, is available. It may also be supported by other models of players (contact PIXILAB for details). Ensure that Caching is enabled in the Display Spot Settings.

Mirroring Service

Supports hardware redundancy for mission critical Blocks applications. The Mirroring Service keeps the content of the backup server in sync with the main server. In case of a failure of the main server, you can immediately switch over to the backup server to keep your installation working while the main server is being repaired or replaced.

Art-Net Recording/Playback

While Blocks can control DMX512/Art-Net lighting fixtures directly, you may want to use a separate lighting console to program more complex lighting effects. Blocks can then record the data stream from the console for later playback directly by Blocks, allowing you to remove the console.

Visitor Data Collection

This optional license enables the ability to track individual visitors, collecting data during the visit, etc.

- For Visitor Spots, it activates the settings found under the *Identity* tab, allowing you to associate a *data record* type with those visitors.
- User Scripts gain the ability to create and use *data records*, such as those associated with Visitor Spots or other forms of identification, such as RFID tags.

Upgrading your License

Your Blocks license can be upgraded by ordering additional, licensable items. The procedure for upgrading your license depends on whether you use a physical or cloud-based license. Follow the instructions shown on the right hand side of the Licenses page to upgrade.

Mirroring Service

This page is available if your license includes the Mirroring Service option. It shows the mirroring status, such as whether the mirroring server is connected, when the last synchronizations were performed, etc.

- **Mirror Connected** states whether the mirroring service connection has been established.

- **Last Full Sync.** Time when the last full sync was performed. This is performed at the moments specified in Blocks' configuration file, or when pressing the "Synchronize Now" button. A full sync means that all pertinent files (including all files inside the Blocks Root directory) have been synchronized.
- **Last Incremental Sync.** Time when the most recent incremental sync was performed. Incremental syncs are performed on a file-by-file basis as you edit content and files in the Blocks editor.
- **Sync Errors** is incremented whenever an error occurs performing a sync operation. If this value keeps increasing after a server start, you should take a look in the mirror service's log file, which can be found under Manage, Status and Log, Archive.
- **Last Sync Error**. Time when the last sync error, if any, occurred.
- **Mirror Clock Offset**. Time difference between the main and mirror server's clocks. If the mirror server isn't connected to the internet (or other clock source), its clock may drift gradually. If this difference is more than a few minutes, you should set the clock manually on the mirror server.
- **Mirror Disk Space** lets you check that there's a healthy amount of free space on the mirror server's disk. If this value dips below 10 GB or so, you should take action to rectifying this, since the mirroring will fail if the server runs out of disk space. That should not normally occur, since files deleted on the main server are then also deleted on the mirror server.

Status and Log

Shows basic health status about your Blocks server, such as the amount of free memory and disk space. Also provides a live view of the server's log, and the ability to download current and previous log files.

Status

The values shown on this page are updated automatically about twice per minute.

- **Free disk** indicates the amount of free disk space on the server volume containing the Blocks data. Take a look at this value every now and then, especially before uploading new media files, to make sure there's enough free capacity. If this value drops below a dozen GB or so, you need more disk space.
- **Free program memory** indicates the amount of unused RAM memory in the server program. It is normal for this value to fluctuate quite a bit, as memory is managed dynamically while the server is running. To see the maximum amount of memory that can be made available at this point, press the "Free Unused Memory" button. If this value drops below 50 MB or so, your server may be running low on memory. If your server has plenty of RAM, the amount of memory allocated to Blocks may be increased by adjusting a server setting. Contact support@pixilab.se for details.
- **Max backlog** indicates the workload of the server. If this value is above 200 or so, you may need a faster server. Alternatively, you may be able to reduce the workload by optimizing some script, task or server setting.
- **Started** shows the date and time the server was last started, thus providing an indication of how long it has been running.

Live Log

Shows server log messages as they appear, including up to 30 recent log messages. To view the complete log, download the "latest.log" file under Archive.

- **Time** and date of the log message. In general, only messages from the current day are visible in the live log view. Download the archived log file for the desired date to view earlier log messages. The archived log also provides time stamps with millisecond accuracy
- **Logger** name, from which this message was generated. You can control the log level for individual loggers using this name under *loggers* in the server's configuration file.

- **Object** states the name of the Blocks object (e.g., a Display Spot) the message pertains to, if available.
- **Message** describes what happened.

Archive

Lists all log files available on the server, with the ability to download the selected log file.

- **HINT:** If you're having trouble with your Blocks server it's often helpful to include the log file for the day in question, along with an approximate time when the problem occurred, when contacting support.

The file name "latest.log" corresponds to what you see under Live Log, but contains the complete log including all older messages logged after the most recent archived log.

Files named "archive," followed by a sequence number and a date, contain log messages for that date. If there's a lot of logging, more than one file may be generated for that date.

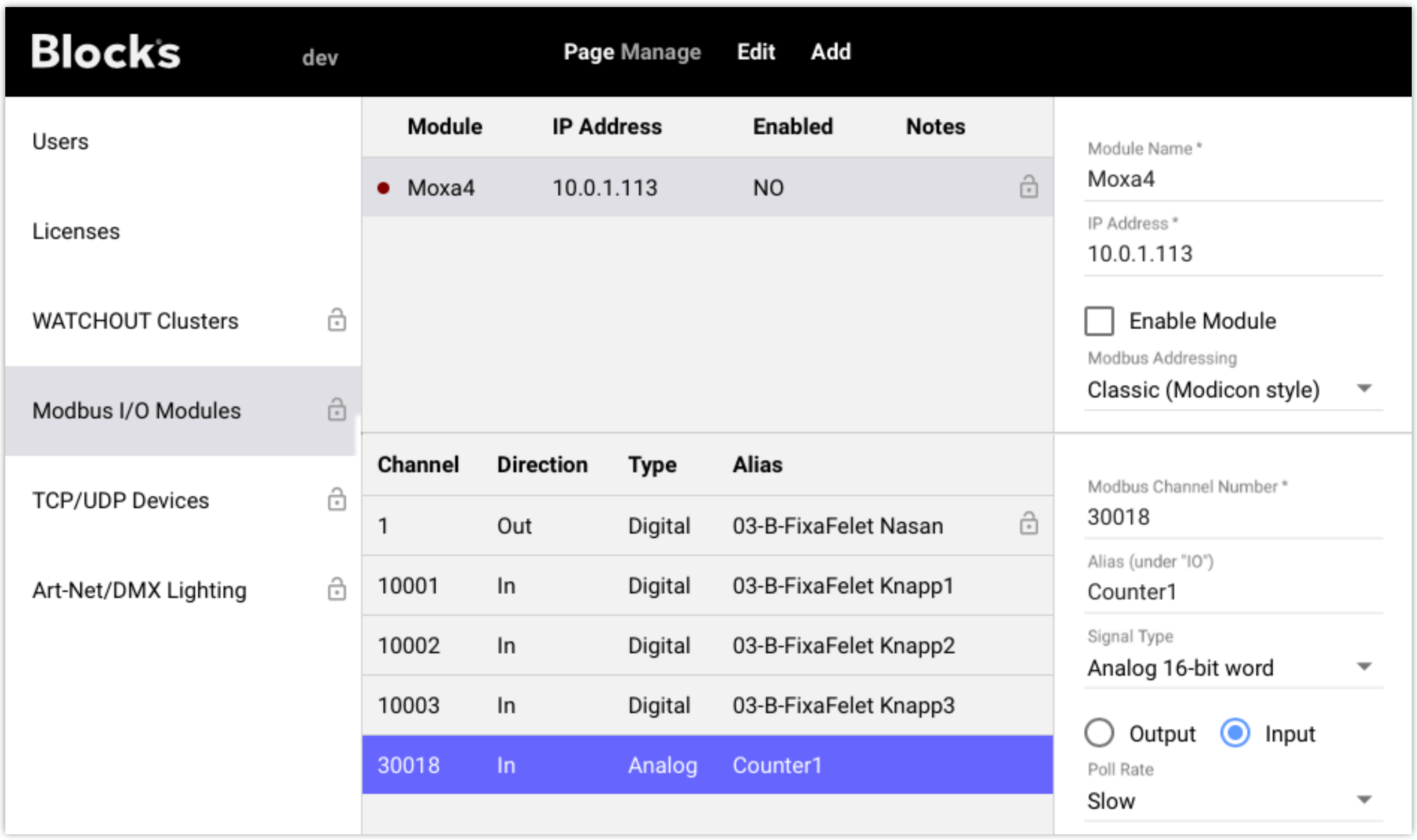
You can control what's logged and how verbose the log is in your server's configuration file.

Modbus I/O Modules

Modbus is an industry standard protocol for communicating with devices such as sensors and relays. Blocks supports the "Modbus TCP" version of the protocol, allowing you to connect such devices over an Ethernet network.

Before you can access a Modbus I/O module from Blocks, you must give it a known IP address. Please refer to the manufacturer's documentation for the proper procedure.

◆ The Modbus TCP protocol has no security built in. Do not use Modbus I/O modules on an open network. If you want to have a public wifi network for mobile guide functions, you must separate that network from the control network in a way that allows guests to *only* access the Blocks server. This can be done, for example, using a properly configured router and VLANs.



Module	IP Address	Enabled	Notes
● Moxa4	10.0.1.113	NO	

Channel	Direction	Type	Alias
1	Out	Digital	03-B-FixaFelet Nasan
10001	In	Digital	03-B-FixaFelet Knapp1
10002	In	Digital	03-B-FixaFelet Knapp2
10003	In	Digital	03-B-FixaFelet Knapp3
30018	In	Analog	Counter1

Module Name * Moxa4
IP Address * 10.0.1.113
☐ Enable Module
Modbus Addressing: Classic (Modicon style)

Modbus Channel Number * 30018
Alias (under "IO"): Counter1
Signal Type: Analog 16-bit word
○ Output [✓ Input]
Poll Rate: Slow

Modbus Module Settings

Once you have the Modbus device properly configured, with a known IP address, add it to the list using the “Modbus Module” command on the Add menu. Give it a name and specify its IP address. This name can then be used to access and control the channels, e.g., from buttons and tasks.

- **Enable Module** lets you temporarily disable a Modbus module so that no commands will be sent to its output channels.
- **Modbus Addressing** controls how channels are addressed (see below under Modbus Channels).
- **Concurrent Queries** controls how many outstanding (not yet responded to) poll requests that Blocks may have on this module. Allowing multiple concurrent queries improves performance. Some devices may not support this, in which case you should set this value to 1 to avoid communication errors.

The red indicator to the left of the module name in the list will turn green when the Blocks server connects to the module.

◆ The Modbus TCP protocol stipulates that modules disconnect from the server regularly, which then turns the indicator red temporarily. This is normal, and should not interfere with the operation of the device, as Blocks reconnects as needed.

Modbus Channels

Most Modbus I/O modules are equipped with screw terminals for connecting the sensors or devices to be controlled. Those screw terminals are addressed by a channel number. Sometimes, the channel number is indicated next to the screw terminal. If not, the module‘s documentation should indicate the mapping between channel numbers and screw terminals.

◆ Some Modbus modules have configurable channel numbers, in which case you may need to set those as well when configuring the module.

Blocks supports two forms of Modbus addresses:

- **Classic (Modicon style)** addressing, with outputs beginning at 1 and inputs at 10000, as shown in the example above.
- **Modern (full 16 bit)**, where any numeric address may be freely assigned to any function.

To add a channel to a module, select the module in the list, then choose “I/O Channel” on the Add menu. Configure the settings for the channel as appropriate.

Modbus Channel Number

Enter the channel number here, as stated in the module’s documentation (or as configured by you, if the channel numbers are configurable).

Channel Name

Alias

While a channel can be accessed under the Modbus module’s name as “chN” (where N is the channel number) or using its given name (see above), it’s sometimes preferable to access the function being controlled by a separate name, independently of the module it connects through. In the illustration above, channel 1 is given the “cleaningLights” alias name. Using this method also makes it easier if you need to move the function to another Modbus module since you can then simply move the alias name, and all your programming will continue to work.

If you add an alias name to a channel, you can later access that channel for programming purposes as “IO.YYY”, where YYY is the alias name given to the channel.

Signal Type

Specifies whether the signal is “Digital” (with on/off states, such as a relay or a pushbutton) or “Analog” (such as a temperature sensor). Furthermore, analog signals may use 16 or 32 bits of precision. Please refer to the documentation of the module being used. If not explicitly stated, it’s usually safe to assume 16 bit precision.

Signal Direction

Specifies whether the channel is an output (for controlling things, such as a relay), or an input (such as a contact closure or a temperature sensor).

Initial State

Specifies the state an output channel will be set to when Blocks is started. For a digital output, enter 1 to set the output's state to *true* on power up. For an analog output, set the desired numeric value. Applies to outputs only.

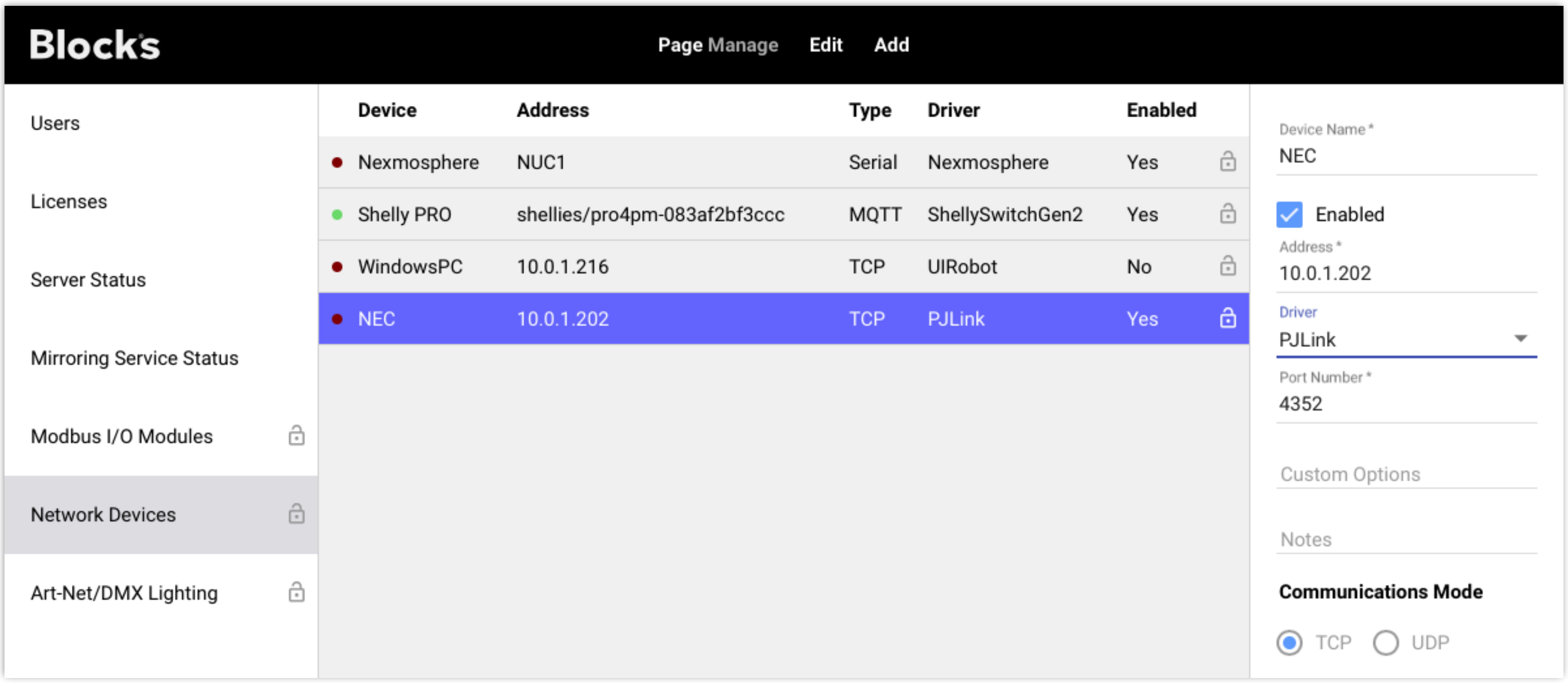
Poll Rate

Modbus requires polling to obtain input data. For inputs that change rarely or slowly, leave the Poll Rate set to Slow. If the input value may change very rapidly, and you need to get those changes sooner, select Fast. Do not select Fast unless actually required, as doing so increases the load on the system.

- Some Modbus modules filter input data. If so, make sure any filter value is set low enough for the highest expected input frequency. For instance, many MOXA Modbus modules have a default filtering time of 100 mS, meaning that input changes faster than 5 Hz may be suppressed by the module. In this case, merely setting the poll rate to *Fast* will not help, as the signal is already filtered by the module. Please check the module's documentation for details.

Network Devices

Many devices can be controlled over the network using standard network protocols such as TCP, UDP or MQTT or through a serial port (e.g., RS232, RS422, RS485). In some cases, all it takes is sending a text string to the device. But in most cases, a device driver should be used, that knows how to talk to the device. Such a device can be controlled from Blocks by adding it under Network Devices on the Manage page.



	Device	Address	Type	Driver	Enabled
●	Nexmosphere	NUC1	Serial	Nexmosphere	Yes
●	Shelly PRO	shellies/pro4pm-083af2bf3ccc	MQTT	ShellySwitchGen2	Yes
●	WindowsPC	10.0.1.216	TCP	UIRobot	No
●	NEC	10.0.1.202	TCP	PJLink	Yes

Device Name * NEC
✓ Enabled
Address * 10.0.1.202
Driver PJLink
Port Number * 4352
Custom Options
Notes
Communications Mode
[✓ TCP] UDP

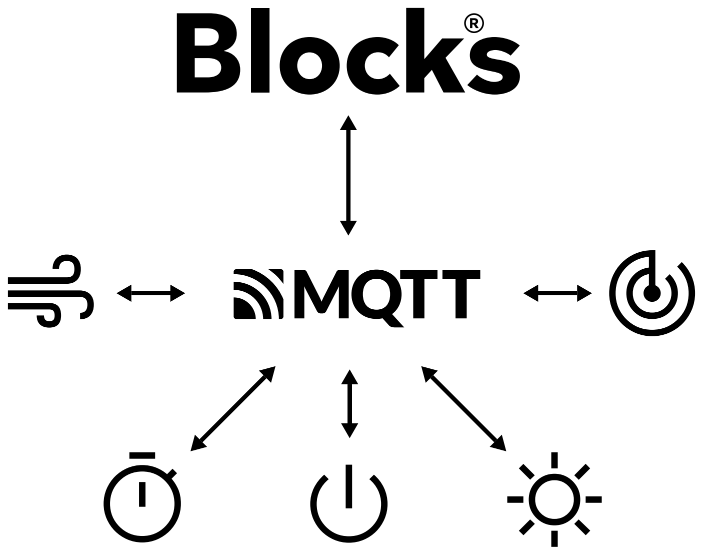
TCP or UDP device

To add a device, choose "TCP/UDP Device" on the Add menu then specify its Name and IP Address. This Address must match the one configured on the device. If using a driver (see below), the Port Number will generally be set automatically. If not, set it manually to the correct IP port for the device. The name given is used to program the device, so use a short but descriptive name. If the documentation doesn't state whether to use the TCP or UDP mode, choose TCP as it is the most common.

When using the TCP communications mode, Blocks may automatically connect to the device. Successful connection changes the color of the status indicator in the list to green. UDP doesn't support connections, and therefore has no indicator.

MQTT device

MQTT is a standard messaging protocol used by many Internet-of-Things devices. MQTT devices connect through an intermediary caller a *broker*. An MQTT broker is included in our Linux-based server image, and is available for free for most operating systems. Blocks communicates with MQTT devices through this broker. Since the MQTT protocol supports encryption and authentication, and generally has low bandwidth needs, it's also suitable for running over the internet, using a cloud-based broker.



To connect an MQTT device to Blocks, first make sure you have a working MQTT broker accessible from Blocks. You may also want to verify that your MQTT device can connect to the broker. This can be done using the "MQTT Explorer" application, mentioned on the web page linked, or some other suitable MQTT GUI/terminal client.

Add your device to Blocks by choosing "MQTT Device" on the Add menu, specifying its name and *topic base*. This topic base is the common, initial part of the MQTT topic path used to address all functions of a device. Think of it as the address of the device, under which sub-topics are then used to address individual device functions.

The **Retain Messages** checkbox sets the default value of the "retain" flag sent to the MQTT broker for outgoing messages. This default will be applied unless a driver explicitly overrides this when sending a message.

Before selecting the device driver, verify that the indicator shown next to the device's name in the list is green. For a driver-less MQTT device, this merely indicates that Blocks has connected to the broker. Finally, select any desired device driver for your device. This device driver knows how to talk to the device, exposing its various capabilities as Blocks properties and functions. Some drivers take additional settings as *Custom Options*. Please refer to the driver for details.

- While using a proper device driver is generally preferable, if one is available, you may also use MQTT without selecting a driver. In this case, you can only *send* data to the device from a Task, by specifying the proper sub-topic and data string. Another options is to use the ConfigurableMQTT driver, which can handle many common MQTT topic-to-property mappings by means of a small amount of JSON configuration data specified in the "Custom Options" field of the driver.

Serial port

Devices using a plain serial data connection for control can be used with Blocks. Such a connection is often labeled RS232, RS422, RS485 or similar. However, modern devices sometimes use a USB port for carrying the serial data, in which case it may not be entirely clear that the device actually uses serial data internally. Your choice of serial connection options include:

1. A device with a USB connector, such as a Nexmosphere sensor hub, can be connected directly to a USB port on a player.

2. A device with a traditional DB9 or DB25 serial data connection (e.g., "RS232") can be connected to a USB port on the player using a USB-to-Serial port adaptor.

3. A device with a traditional serial data connection can be connected through a network-to-serial interface.

Option 3 is essentially the same as a TCP network device, since that is what the network-to-serial interface is. Option 1 and 2 above are suitable if you have a Display Spot based on PIXILAB Player version 6.0 or later nearby. To use this option:

1. Connect the device or adaptor to a USB port on the Display Spot (e.g., the NUC driving the display).
2. Add the device to Blocks by choosing "Serial Device" on the Add menu under Network Devices on the Manage page.
3. Enter the name of the Display Spot into the "Connected through Spot" field. If it's in a group, prefix the name with the name of the enclosing group(s), separated by periods, like this: TopGroup.InnerGroup.MySpot
4. Select a suitable driver (if any), or – if you just need to send some raw data to the device – specify the Baud Rate used by the device.

Control the device by using its properties and functions. If a device driver is being used, the set available is determined by the driver. If no driver is being used, only the functions used to send raw data are available

Serial device drivers are conceptually similar to TCP device drivers. As a matter of fact, they are so similar that a single device driver can serve both purposes. This is particularly useful for devices that can connect either through a network-to-serial interface (in which case Blocks will consider it a TCP network device) *or* a USB-to-serial interface (in which vase Blocks considers it a Serial device). As an example where this is the case, see Blocks' Nexmosphere driver.

Device Driver

Some devices have more complex protocols, or require specialized handshaking or other considerations for establishing reliable communication. Such devices can be hard to control by just sending command strings. Furthermore, the syntax used by some protocols isn't easy to understand and get right.

To overcome such issues, specialized "device drivers" can be provided. This is a piece of software that has detailed knowledge of the device's communication protocol, and knows how to talk to it in a reliable way. Device drivers are available for standard protocols, such as those commonly used by video projectors (e.g., PJ-LINK, used by many projector manufacturers).

An added benefit of a device driver is that it can expose higher level functionality, rather than the details of the underlying communication protocol. For instance, you may want a "power" property for turning a projector on or off, or perhaps an "input" property for switching video inputs. The driver typically exposes such properties, which can then be directly linked to buttons and tasks for control.

Once a device driver has been installed on the server, it can be selected on the "Driver" menu in the settings pane. New drivers are developed by PIXILAB and partners as needed. Contact PIXILAB for the latest information on available drivers, or check out PIXILAB's github page where we keep most drivers. Note that some less frequently used drivers are stored in a *driver-archive* directory. To activate such a driver, move it into the *driver* directory and restart your Blocks server. If a driver doesn't appear in the Blocks editor, or doesn't work as expected, check Blocks' log file, located at *logs/latest.log* inside your Blocks root directory, as it often provides clues on what's amiss.

Custom Options

This settings field can contain arbitrary data, which is made available to the device driver. It can be used for driver-specific settings, often using JSON syntax. Refer to the driver's documentation for details.

Art-Net/DMX-512 Lighting

Theatrical lighting fixtures can typically be controlled using the DMX512 industry standard protocol. Some other lighting systems have “bridges” that allow them to also be controlled by DMX512. DMX512 is at its core a simple protocol, allowing you to set or fade the value of up to 512 channels. These may be used to directly control the intensity of a large number of lights. The channels can be combined in various clever ways to control complex lighting fixtures, such as “moving heads” with pan/tilt operation, and numerous other effects and functions.

While DMX512 stipulates also the hardware wiring used to connect the lights, DMX-512 control data is often sent over standard Ethernet wiring, where it may share the network with other functions. This method uses the Art-Net protocol to “tunnel” the DMX data over the regular network. The Ethernet network connects to dimmers and other devices directly compatible with the Art-Net standard. Alternatively, it can be bridged using an Ethernet-to-DMX adapter, available from numerous vendors.

Blocks can control such lighting fixtures and dimmers using the Art-Net protocol. This can be done either interactively – using buttons and sliders – or in a pre-programmed way, using Tasks. For fixtures equipped with DMX512 XLR connectors, an Ethernet-to-DMX adapter, as mentioned above, is required.

Another way of controlling lights from Blocks is by recording a sequence of programming, for example originating from an external lighting console. Such a recording can then be played back by Blocks. This method is particularly useful for more complex lighting effects, then created using a professional lighting console.

Adding a Fixture

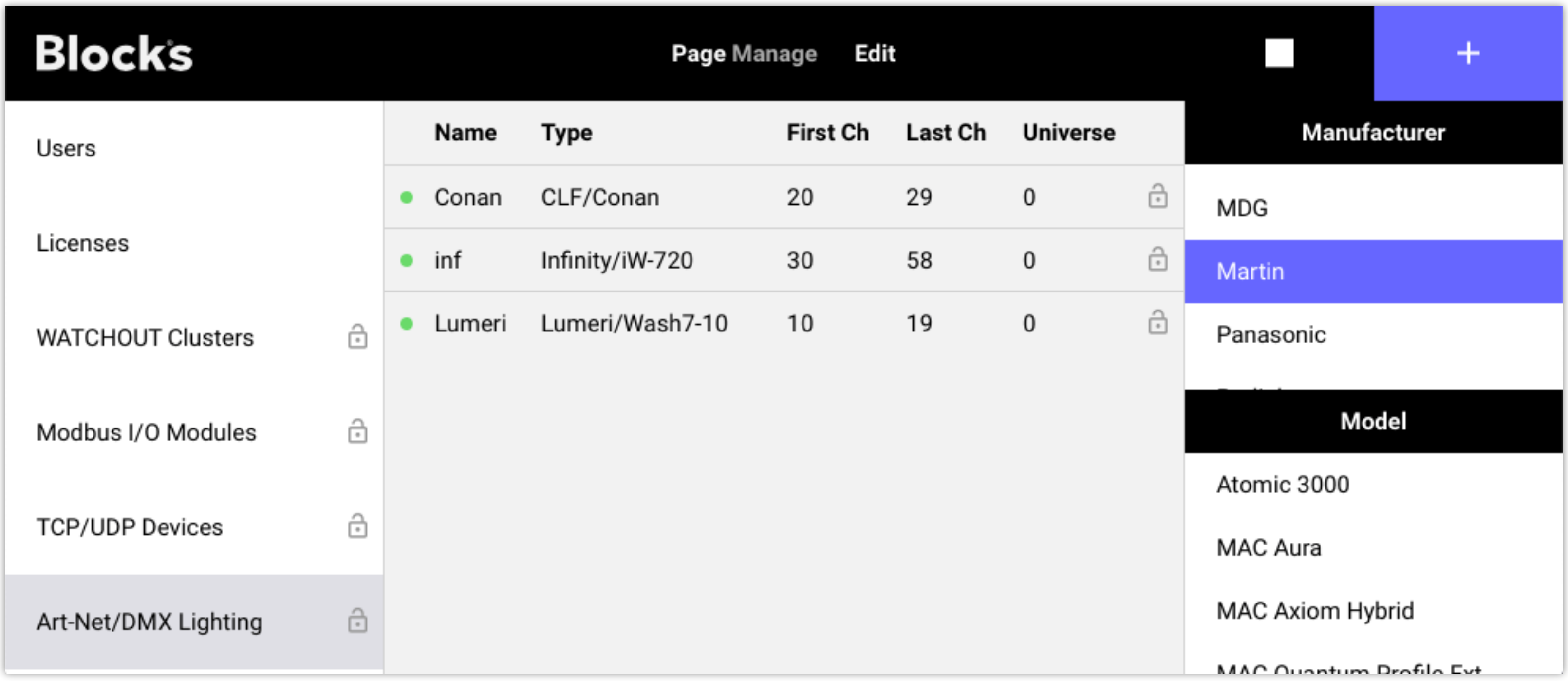
To add a fixture to be controlled, do as follows:

1. Choose “Art-Net/DMX lighting” on the Manage page, as shown below.

2. Click the + button in the top right hand corner.

3. Drag a fixture model from the Model list into the main area of the window.

Blocks comes with a set of generic fixture definitions (found under the “Generic” manufacturer). Additional fixture definitions are found on the PIXILAB github page, along with instructions on how to install them, as well as how to create your own, if required.



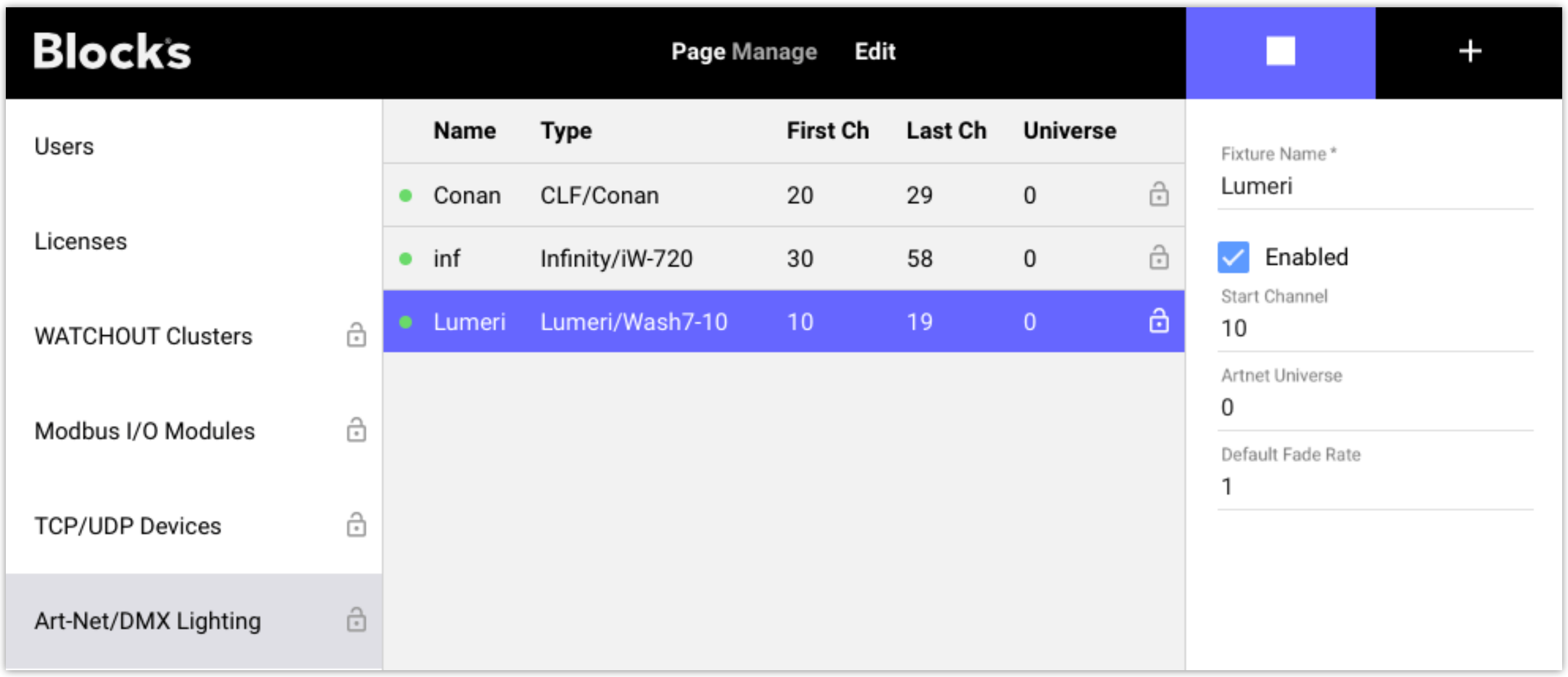
Name	Type	First Ch	Last Ch	Universe
Conan	CLF/Conan	20	29	0
inf	Infinity/iW-720	30	58	0
Lumeri	Lumeri/Wash7-10	10	19	0

For basic, single-channel control, choose “Desk Channel” under Model after selecting “Generic” under Manufacturer. Drag the “Desk Channel” to the main area of the window to add this channel. A few other basic definitions are found under “Generic”, such as “RGBD Fader” for controlling a color fixture with Red, Green Blue and Dimmer channels, or the “Pan Tilt 16-bit” for controlling a moving head with two high resolution channels for pan and tilt motion.

Fixture Settings

After adding a fixture definition, click the square in the top right hand corner to adjust its settings, as shown below. Type a name describing what the fixture illuminates, or that makes sense to you in some other way, as this name will subsequently be used to program the light. Specify the start channel (DMX512 channel number), corresponding to the base channel setting of the fixture. If the fixture uses more than a single channel, additional channels are mapped consecutively following this first channel. You can see the range of channels consumed by the fixture in the list, as indicated by "First Ch" and "Last Ch". The number of channels used by a fixture is specified by its fixture definition.

The simple "Generic Desk Channel" fixture uses only a single, 8 bit channel, as indicated by the "First Ch" and "Last Ch" being the same. The "Lumeri" fixture selected below uses ten channels, numbered 10 through 19, with 10 being its base channel.



Name	Type	First Ch	Last Ch	Universe
Conan	CLF/Conan	20	29	0
inf	Infinity/iW-720	30	58	0
Lumeri	Lumeri/Wash7-10	10	19	0

Fixture Name *
Lumeri

✓ Enabled

Start Channel
10

Artnet Universe
0

Default Fade Rate
1

Artnet Universe

Due to the higher bandwidth of Art-Net/Ethernet compared to the slower DMX512 bus, it is possible to tunnel several DMX512 buses over a single Ethernet wire. These are referred to as "universes" in the Art-Net protocol. You specify which Art-Net universe to use for a fixture using the "Artnet Universe" setting, which must match the corresponding setting in the receiving end.

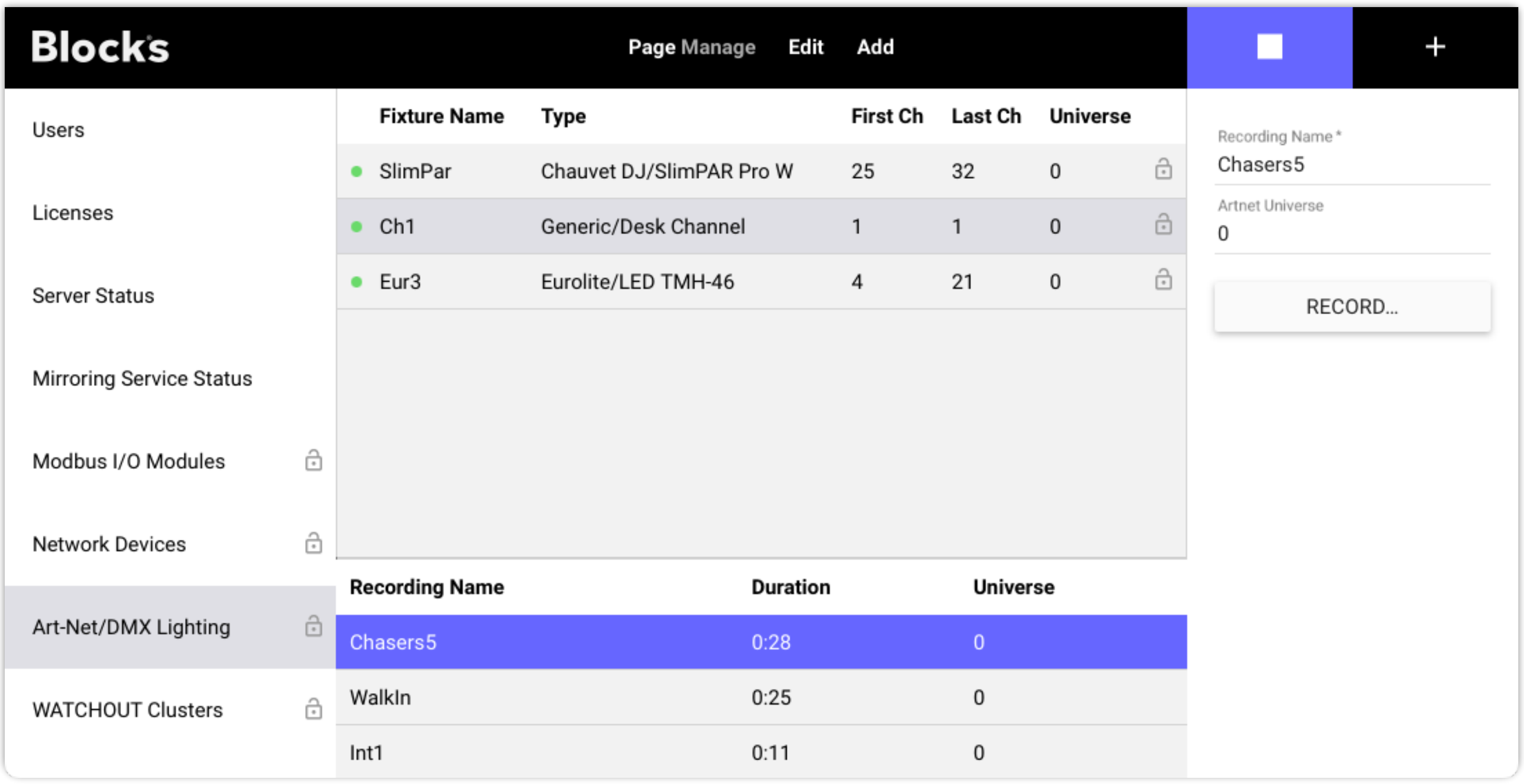
- While the channel number is 1-based, the universe is 0-based. Some devices may have a different convention here, resulting in "off-by-one" issues. Thus, if your device doesn't respond, you may want to try one universe above/below the number set in Blocks, to see if that makes it work.

Default Fade Rate

This setting controls how fast a channel will reach the target value when controlled from a button. When controlling the channel from a task, you can explicitly specify the fade time in each task statement, without having to rely on this setting.

Set this value to 0 to set the target value immediately when pressing a button, rather than using a gradual fade.

Art-Net Recording



Fixture Name	Type	First Ch	Last Ch	Universe
SlimPar	Chauvet DJ/SlimPAR Pro W	25	32	0
Ch1	Generic/Desk Channel	1	1	0
Eur3	Eurolite/LED TMH-46	4	21	0

Recording Name *
Chasers5

Artnet Universe
0

Recording Name	Duration	Universe
Chasers5	0:28	0
WalkIn	0:25	0
Int1	0:11	0

Blocks can record an Art-Net data stream in real time, for example originating from an external lighting console.

1. To record from an external lighting console, first connect the console using Art-Net to your Blocks network. Configure the console to either unicast to the IP address of your Blocks server or to broadcast (using either a global broadcast address of 255.255.255.255 or a subnet-specific broadcast address matching your Blocks network)

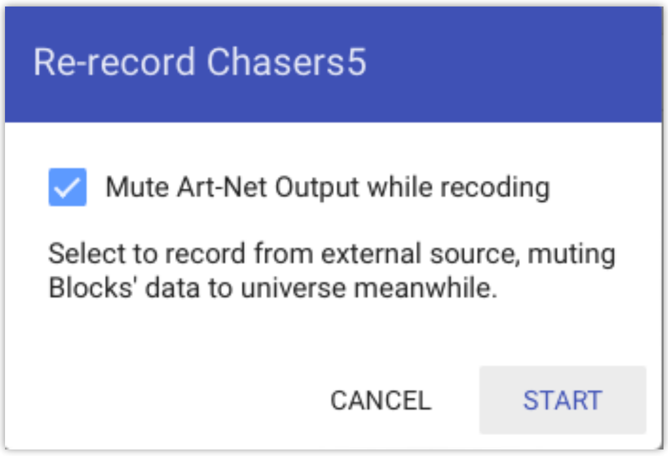
2. On the Add menu, select Recording to add a new recording. Name the recording as desired, and specify the Art-Net universe number on which the console sends its data. Note that the first Art-Net universe in Blocks is numbered 0, while some other systems may use 1-based universe numbers.

3. To make a recording (or to re-record an existing recording) press the “Record...” button to open the recording dialog box.

4. To record from an external console, make sure that the “Mute Art-Net Output while Recording” is selected. Unselect this if you instead want to record Art-Net data originating from Blocks itself.

5. Press START to begin recording.

6. 6. To stop a recording in progress, select the recording then click the “Stop Recording” button.


Re-record Chasers5

✓ Mute Art-Net Output while recoding

Select to record from external source, muting Blocks' data to universe meanwhile.

To play back an Art-Net recording, do either of the following:

- For free-running playback, set the *playing* property of the recording to true using a Button or a Task.

- For synchronized playback, use a Task to call the *startSync* function of the recording, passing it the path to its synchronization time source and an optional time offset. Call *stopSync* to terminate synchronized playback.

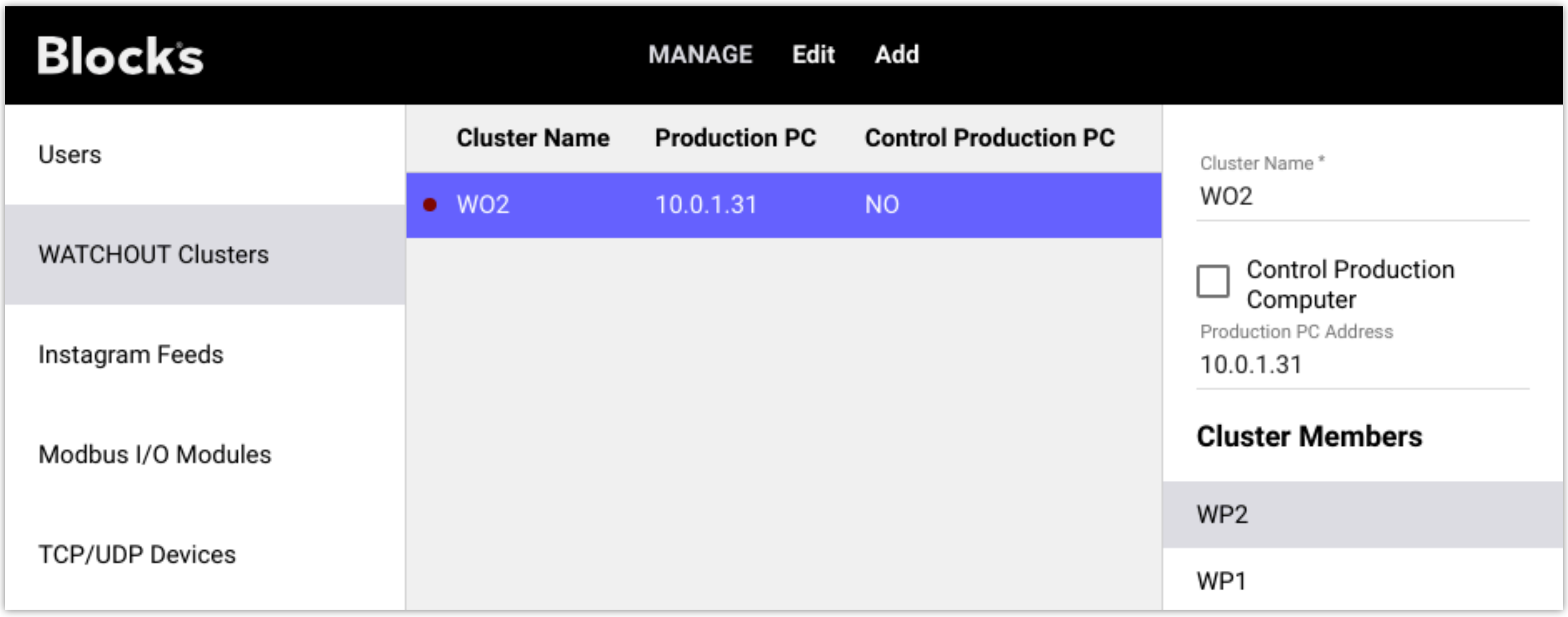
◆ **NOTE**: Art-Net recording and playback requires a separate licensing option.

WATCHOUT

WATCHOUT can be used with Blocks for advanced multi-projection or mapping purposes. A WATCHOUT systems consists of display computers, and (optionally) a production computer. The production computer is typically only used during show preparation, and can then be removed. However, in some cases, it's left as part the system in order to facilitate quick changes.

- **NOTE:** In WATCHOUT version 7, only the main computer ("director") can be controlled, since it doesn't have the same cluster concept as previous versions. It must be addressed by its IP address. Furthermore, functions that have no counterpart in that version, such as stand-by and layer conditions, can not be controlled. Auto-discovery and remote power on/off of the computers in the system is not supported.

Add a WATCHOUT subsystem using the "Add" menu. If you plan on keeping the production computer as part of the system, enter its IP address in the "Production PC Address" field. If not, leave that field empty. Alternate between controlling the production computer or the display cluster using the "Control Production Computer" checkbox. The red dot in the list turns green to indicate a successful connection. Blocks will automatically retry failed connections until it succeeds.



Cluster Name	Production PC	Control Production PC
● WO2	10.0.1.31	NO

Cluster Name * WO2

☐ Control Production Computer

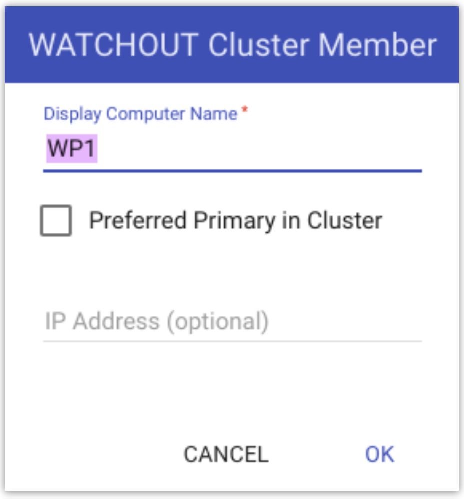
Production PC Address 10.0.1.31

WP2

WP1

WATCHOUT version 6 specific options

The name you give should match any "Cluster Name" specified in the Preferences dialog box in WATCHOUT 6. Using the same name, as well as display computer names (see below), allows Blocks to find the cluster on the network without having to specify fixed IP addresses for each display computer.



Display Computer Name * WP1

☐ Preferred Primary in Cluster

IP Address (optional)

Add display computers using the "Add Member" command on the "Add" menu. If you're controlling the production computer, this isn't strictly necessary, as all control will then go through the production computer anyway. But if you later want to remove the production computer and control the display computers directly, you should add those as well to support that mode of operation. When adding display computers, you must give each computer a name. This name should match any name given to the display computer in the WATCHOUT production software.

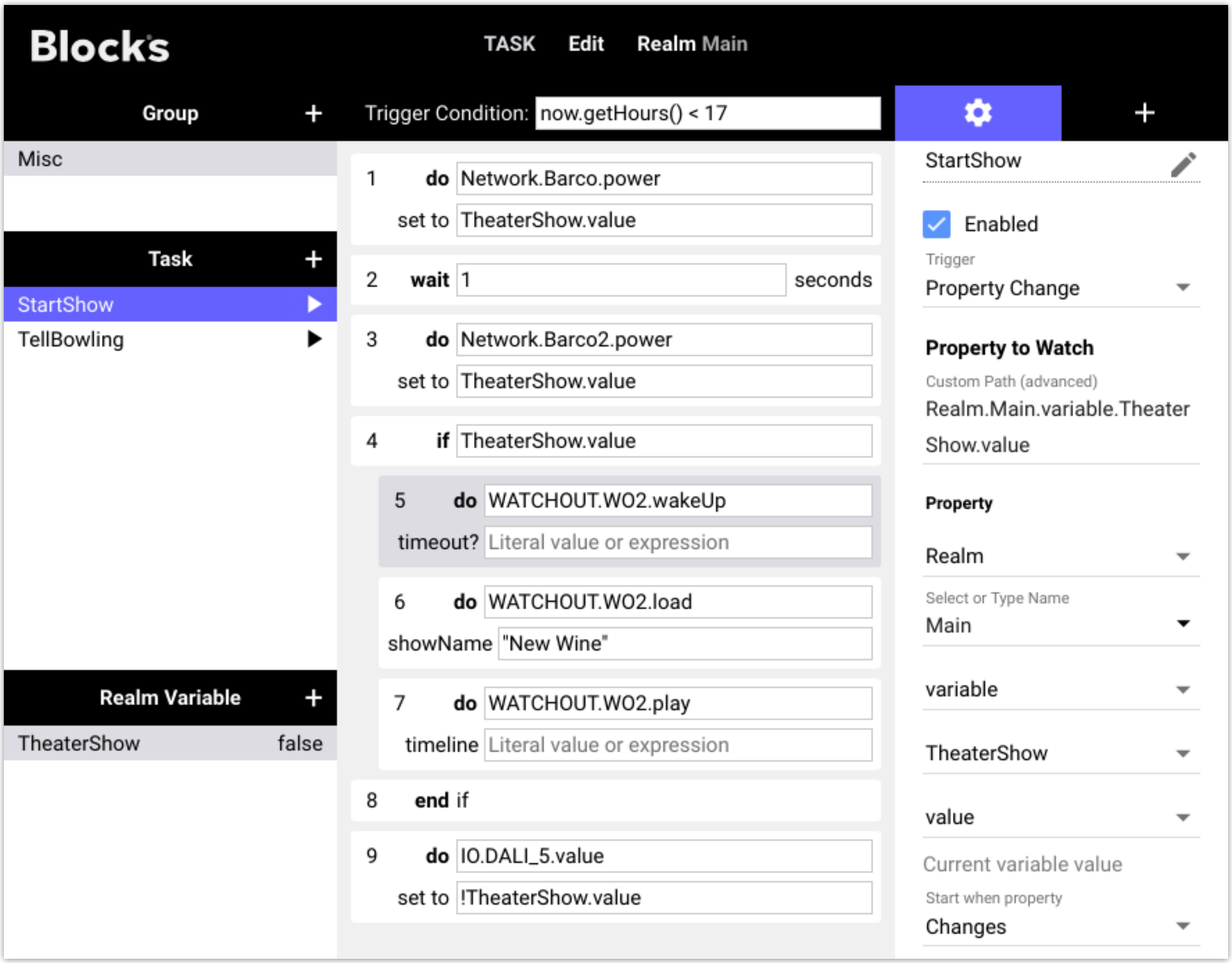
Optionally, you may specify an IP address for each display computer. Use this method if you use fixed IP addresses for the display computers in your WATCHOUT show. You may also designate one display computer to act as the "primary" in the cluster, which tells it to act as the master within that cluster. Usually, you don't care which computer becomes the master, in which case you can leave this option unchecked.

Controlling WATCHOUT

Once added to Blocks, you can control WATCHOUT using buttons, sliders and tasks or use it as a synchronization time source. You can also use events, such as a timeline reaching its end, to trigger tasks.

13. TASK

Basic control needs can often be met by directly linking buttons and other controls to the properties being controlled. That method can be used to turn on the power of a projector using a button, or to set the volume level using a slider. In other cases, however, you may need to control *multiple* functions together – such as turning on *several* projectors *and* setting the light level in the room – or send commands in a particular sequence. Such more complex control scenarios can be accomplished using *tasks*.



Trigger Condition: now.getHours() < 17

```
1    do Network.Barco.power
 set to TheaterShow.value
2  wait 1 seconds
3    do Network.Barco2.power
 set to TheaterShow.value
4    if TheaterShow.value
   5    do WATCHOUT.WO2.wakeUp
   timeout? Literal value or expression
   6    do WATCHOUT.WO2.load
   showName "New Wine"
   7    do WATCHOUT.WO2.play
   timeline Literal value or expression
8  end if
9    do IO.DALI_5.value
 set to !TheaterShow.value
```

TheaterShow false

StartShow

Enabled ✓

Trigger: Property Change

Property to Watch

Custom Path (advanced): Realm.Main.variable.TheaterShow.value

Property

Realm

Select or Type Name: Main

variable

TheaterShow

value

Current variable value

Start when property: Changes

A task that turns projectors on or off and runs a presentation.

A Blocks server can handle multiple tasks at the same time. Tasks are organized into Realms and then into Groups. Tasks can communicate with Spots and other devices added on the Manage page (such as I/O modules and network devices). Tasks can also communicate with each other, either by triggering one another, or by posting notes in "mailboxes" known as *Realm Variables*.

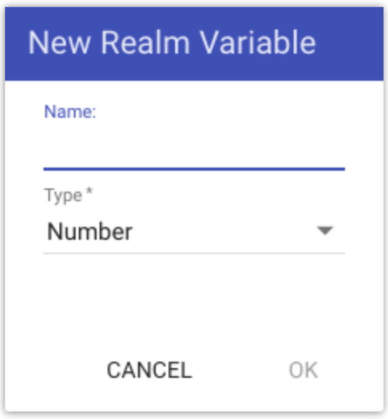
◆ While these *tasks* can handle most control scenarios, Blocks also supports *advanced scripting* based on a more formal programming language. See "Advanced Scripting and Device Drivers" for more information on this option.

Realm

At the top level, tasks are organized into *realms*. In basic applications, a single realm is all you need. Use multiple realms in larger systems, where you may need to control several more or less independent areas, such as different rooms in a building. While each room may have numerous things to control – often in an inter-related manner, separate rooms may have little or nothing to do with each other. In this case, it makes sense to manage each room as a separate realm.

Select the current realm using the Realm menu in the menu bar. This menu is also used to add and remove realms. When you select a realm on this menu, the Group, Task and Variable lists show those objects in the selected realm.

Realm Variables



Type: Number

In addition to *groups*, and the *tasks* they contain, a realm can also contain shared *variables*. Those variables are directly accessible from all tasks in this realm. They can be used to communicate system state between tasks. They can also be linked to buttons and sliders, providing interactive control over such system state.

To add a Realm Variable, click the plus sign in the list heading. Give the variable a name and specify its type. The name of a variable must begin with a letter, and may only contain letters, numbers and the underscore character. Specifically, it may not contain spaces.

Chose the name wisely, indicating the variable's purpose. This name will be used to access the variable from within tasks as well as for connecting it to buttons and other panel controls.

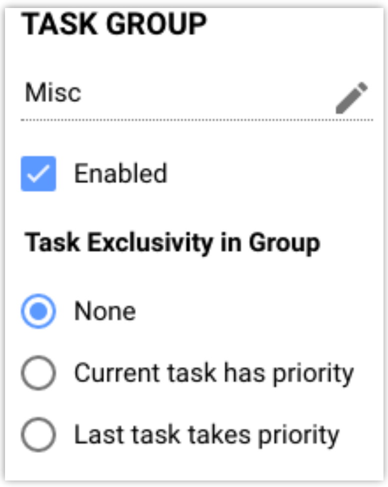
Variable Types

A variable has a fixed type, dictating what kind of data can be stored into it, as well as how it interacts with buttons and other controls.

- **Number.** Contains a numeric value, which may have a fractional part and may be negative.
- **String.** Contains arbitrary characters. Literal character strings, when used in tasks, must be enclosed in single or double quotes.
- **Boolean.** Is either true or false.

Once created, you can configure the variable further in the settings pane. You can specify an *Initial Value*, which the variable will assume when the system is started. To manually change the value, enter a new value in the *Set Value* field then press enter. For a numeric variable, you can specify its minimum and maximum allowed value, as well as whether it can contain fractions or must be a whole number. Those constraints will be enforced when setting the variable through a panel control or a task. If you leave both the minimum and maximum at zero, no range constraints will be applied.

Group



Misc

☑ Enabled

● None
○ Current task has priority
○ Last task takes priority

Tasks are further organized into *groups*. Add a group by clicking the plus sign in the Group header. This grouping has two purposes:

- *Organizational*, allowing you to group related tasks together.
- *Functional*, forcing tasks within a group to be mutually exclusive.

Select a group to change its settings, as shown to the right. To disable an entire group uncheck the "Enabled" checkbox. Disabled groups are shown in italics. Their tasks will not run.

Task Exclusivity

While multiple tasks *can* run concurrently, that's not always what you want. In some applications, you may have a group of mutually exclusive tasks, of which only a single one can be in effect at any given time. For instance, you may have tasks that set a room to various presentation modes. In one mode, the projection screen should be rolled down, in another it should be hidden. It may take some time to switch modes, as it takes a while for screens to move and projectors to wake up. You can account for these delays using "wait" statements, or there may be implied waits caused by some lengthy actions, making commands happen in the right order and with proper timing.

While such a potentially lengthy task is in progress, you may change your mind about the desired room mode. If you then press another button to select that mode instead, things can get out of hand as you now have *two* tasks, both attempting to set the room's mode, essentially fighting each other.

Task exclusivity can help avoiding such potentially chaotic situations by ensuring that only a *single* task within a group of tasks can run at any given time. This is done by selecting one of the exclusive modes:

- **Current task has priority.** A new task in this group can't be started while another one is running. The current task must first run to completion.
- **Last task takes priority.** When you successfully trigger a task in this group, any already executing task in the group will be terminated.

◆ The "Last takes priority" mode may sometimes have unexpected consequences, since any previous task may be interrupted halfway through. You need to make sure that tasks in such a group set all relevant system state explicitly, without relying on other tasks always finishing cleanly.

Task

A task combines a list of statements with a trigger. The trigger controls *when* the task is performed, and the statements inside the task determine *what to do*. As you see in the illustration at the beginning of this chapter, the statements take up the main area, while the trigger is shown on the right hand side when a task is selected in the Task list.

Learn more about how to program tasks in the Blocks wiki: https://pixilab.se/docs/blocks/tasks

Creating and Naming Tasks

Add a task by clicking the plus symbol in the Task list header. Give the task a descriptive name. While you may use any character in a task name, some programming involving tasks becomes easier if you begin the name with an alphabetic character and then only use characters, digits and underscore characters in the name.

Enabled

A task also has an Enabled checkbox. This allows you to disable individual tasks. Note that this is in addition to the similar checkbox in the enclosing Group. Both checkboxes must be selected for a task to run. Disabled tasks are shown in *italics* in the Task list.

Abort on Error

If checked, any error that occurs in a task will cause the task to be terminated immediately, and not execute any following statements. An error message will be logged for the task, indicating what went wrong, This is the default behavior. In some cases, you may want the task to continue running the following statements even if an error occurs. If so, uncheck this option. This will log the error, but will not interrupt the task.

Triggers

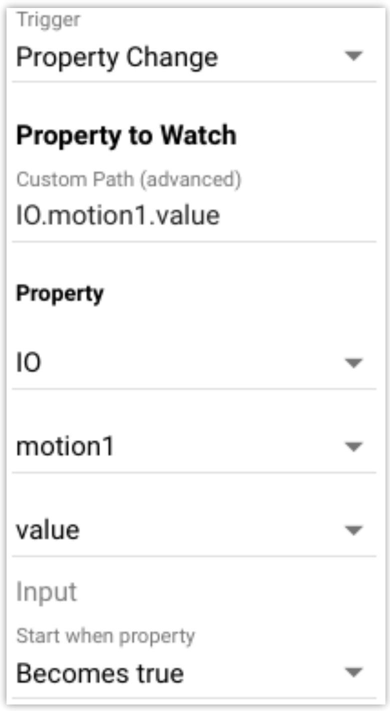
There are a number of ways by which a task can be triggered:

- Manually, by clicking the small triangle next to the task's name. Shift-click to start the task regardless of any Trigger Condition.
- A system property state change. This can be a Spot starting to play, a sensor detecting motion, or the volume being changed.
- Time of day, allowing you to schedule tasks to be performed at a certain time, or repeatedly at some interval.
- Server Startup, runs once when the server is started.

Starting a Task Explicitly

In addition to those two methods built into the task itself, you can also start a task by linking a panel button directly to the "running" state of the task, or from another task using a **do** statement (described later in this chapter). You can use this explicit method regardless of the Trigger mode of the task. Set the Trigger mode to "None" if you *only* want the task to start in this way.

Property Change Trigger



Trigger
Property Change

Property to Watch
Custom Path (advanced)
IO.motion1.value

Property
IO
motion1
value
Input

Start when property
Becomes true

To trigger a task when some system state changes, first select "Property Change" on the Trigger menu, then specify the property to watch using the fields below. The example shown to the right uses the value from a motion sensor, connected through a Modbus I/O Module's input channel as the trigger.

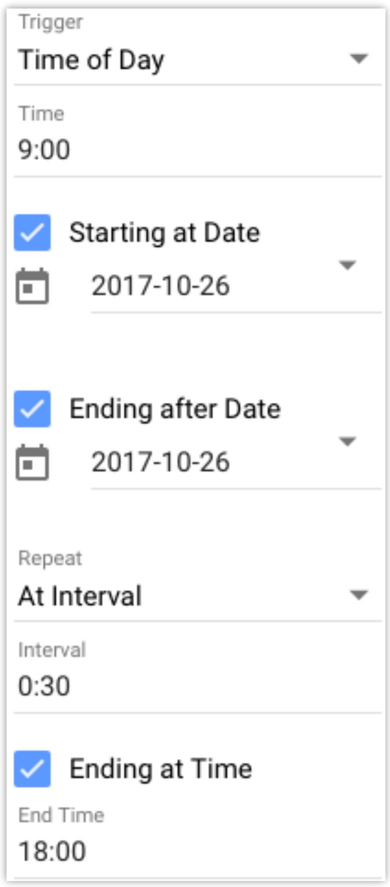
- ◆ The Alias feature of the Modbus channel was used to give it the name "motion1". This makes it easier to program than using the default name based on the Modbus channel number (e.g. "ch10004").

After choosing the property to use as the trigger, you can further qualify the trigger using "Start when property":

- **Becomes true**, triggers the task when the motion sensor senses motion (i.e., the output from the sensor is closed).
- **Becomes false**, triggers the task when the sensor no longer senses motion (i.e., the output from the sensor is released).
- **Changes**, triggers the task whenever the value changes, regardless of direction. This is particularly useful with numeric or string values, and can be further qualified using the Trigger Condition.

If "Becomes true" or "Becomes false" is selected for a numeric property, the value 0 is considered *false* and any other value is considered *true*. If connected to a string property, an empty string is considered *false*, while a non-empty string is considered *true* (this applies even if it contains a single space).

Time of Day Trigger



Trigger
Time of Day

Time
9:00

✓ Starting at Date
2017-10-26

✓ Ending after Date
2017-10-26

Repeat
At Interval

Interval
0:30

✓ Ending at Time

End Time
18:00

To trigger a task at a specific time, first select "Time of Day" on the Trigger menu, then specify the start time and other constraints using the fields below.

The first Time field specifies the time of day when to first start the task. The time is given in 24 hour format, as hours and minutes.

Next you can specify a starting date; useful if you want to set up a task ahead of time to begin at a later date. Likewise, you can also specify an ending date.

- ◆ The Ending date specifies the last day on which the task will trigger. To run a task on a single day only, select that date as both the Starting and Ending dates.

A task triggered at time of day can run either once or repeatedly over a specified time. You specify this behavior using the "Repeat" menu:

- **None** runs the task just once, when triggered. Useful for duties that are only performed in the morning or in the evening, such as shutting power down.
- **At Interval**, restarting the task at the specified interval in relation to its starting time. E.g., if the starting time is 9:00 and the interval is 0:30, it will also be started at 9:30, 10:00, 10:30, and so on, until the End Time is reached.
- **Continuously**. The task will run repeatedly, throughout the day, until the specified End Time. Use a **wait** statement in the task to reduce its frequency.

When you specify a repeating task, you can specify an ending time after which no further repeats will be started that day.

Trigger Condition

You may want to apply further constraints to the triggering of the task than what can be expressed using the Trigger settings. For instance, when using the "Changes" mode of a Property Change trigger, you may want to consider the current value, to only trigger the task if it is above a certain level. Or, when using the Time of Day trigger, you may want to limit the task to certain weekdays.

The "Trigger Condition" field above the list of statements can be used to apply such constraints. The syntax used in this field is based on JavaScript, so having some familiarity with that programming language will help you making the most out of this. The expression you enter here has implicit access to a variable named *trigger*. Depending on what triggers the task, this variable contains either of the following:

- If the triggering mode is "Time of Day"; a JavaScript Date object. This object provides methods to determine the current time, date, weekday , etc.
- If the triggering mode is "Property Change"; a number, boolean, string or object indicating the property's new state.
- For the "Server Startup" triggering mode, the value of the *trigger* variable is undefined. You can use the predefined variable *now* if you need to know what time it is.

For example, if the triggering mode is "Property Change", and the value watched is a volume level expressed as a fractional number between 0 and 1, the following condition would trigger the task only if the current value is above 0.6:

```
trigger > 0.6
```

If the triggering mode is "Time of Day", you can write a condition like this to run the task on all days except Sundays:

```
trigger.getDay() != 0
```

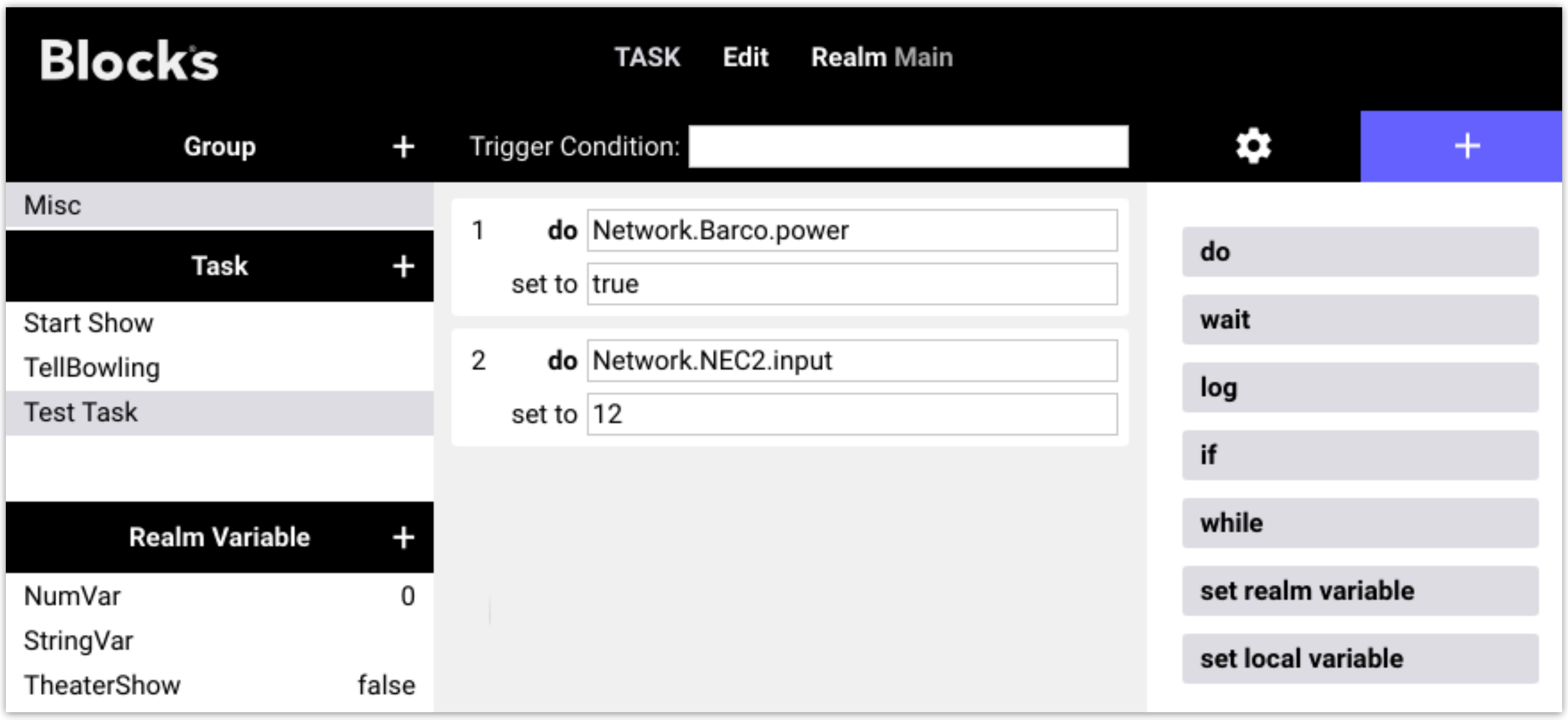
This invokes the getDay() method on the trigger Date object to get the day of the week, as a number 0…6 (Sunday through Saturday). It then uses the "not equal to" operator to compare this to the number 0 (which represents Sunday). The task will *only* be triggered if the condition evaluates to *true*, which this one will do all days of the week except Sundays.

Status, Errors and Logging

A running task is indicated by a number shown to the right of the name in the Task list. This number indicates the current line number in the task, and can be used to track the progress of tasks. This number is only shown while the task is running. For short tasks, this status can be hard to see, as the task may only run for a fraction of a second. You can slow down the task by inserting **wait** statements at desired positions. Alternatively, you can use **log** statements to explicitly display text and values from within tasks. Such log messages show up in a log pane, appearing automatically below the statements when needed. This log pane will also list any errors encountered while running the task.

Statements

Statements determine what a task will do, and how. The statements appear in the center of the Task page when you select a task in the Task list, as you can see in the illustration below. To add a statement, click the plus sign in the top right hand corner, then drag a statement from the right hand column into the list.



1 do Network.Barco.power
set to true

2 do Network.NEC2.input
set to 12

Drag new statements from the right hand column into the statement list.

Drag statements in the list to change their order. To move statements between tasks, first select the statements then choose Cut/Copy/Paste on the Edit menu. Select multiple statements by Shift-clicking or Command/Control-clicking.

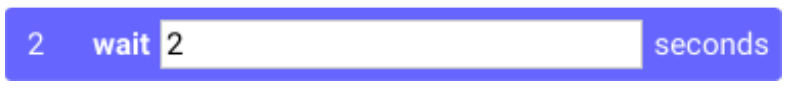
Statement Settings

The settings associated with a statement can be edited directly in the statement's body. Alternatively, for some settings, choose from the menus appearing in the settings pane to the right after selecting a statement.

Statement Parameters

Some statements are very simple, like the wait statement which takes just a single value. Others are more complex, accepting multiple values, called *parameters*. Those use expressions similar to what you can use in the Trigger Condition field, also based on JavaScript syntax. Some basic familiarity with the JavaScript language is advantageous when programming tasks.

Parameters sometimes have a required type. For instance, the single parameter to the wait statement must be a number, specifying how long to wait, in seconds. To specify the value for a parameter, use either a literal value or an expression.



2 wait 2 seconds

*A **wait** statement using a literal, numeric value.*

In some cases, the value isn't known ahead of time, but needs to be calculated while executing the task. This is done using an *expression*, which is akin to the formulas you use in a spreadsheet. For instance, you could specify the value in the above statement like this:

```
1+1
```

This uses an *addition operator* to calculate the value. But as this expression only uses literal constants, it doesn't really have any advantage over just typing the number 2. To do something more interesting, you need to include some *variables* in the expression. For instance, let's assume you want to make the task wait at this point for 4 seconds before noon and 2 seconds after noon. That could be accomplished with the following expression:

```
now.getHours() >= 12 ? 2 : 4
```

That expression first obtains the hour component of the current time, using "now.getHours()". Here *now* is a global variable that always provides the current time, as a Date object. The getHours() method extracts the hours component, as a number 0…23. This is then compared to the value 12. If its greater than or equal to 12 (the ">=" operator) it yields the value 2, otherwise it yields 4. This last step uses a JavaScript operator that returns either of two values depending on the outcome of a boolean (true/false) expression. Learn more about JavaScript operators here:

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Expressions_and_Operators

What you can Use in Expressions

As you saw in the example above, the word *now* can be used to obtain the current time, as a Date object. Math and Number provide some useful constants and utility functions related to numbers and mathematical operations. Literal (constant) values are specified in either of the following ways, depending on their type:

- *23* or *-345.22* for numbers.
- *true* or *false* for booleans.
- *"Will"* or *'Smith'* for string (text) literals. You can use either 'single' or "double" quotes to enclose the string. These must all be 'straight' quotes, not their 'curly' cousins.

◆ If you're new to programming, it's easy to forget the quotes around string literals. If you forget, and type *Will* instead of *"Will"*, things won't work as expected. Whenever in doubt, use a log statement to learn what result an expression produces.

You can get the value of a *local variable by* just stating the name of the local variable (see "set local variable" later in this chapter).

You can access to the value of a *realm variable* like this:

```
NumVar.value
```

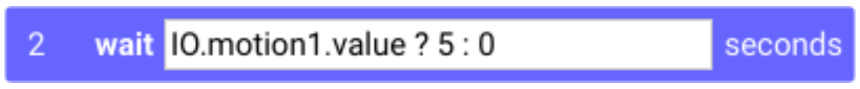
where NumVar is the name of the realm variable (see "Properties and Functions").

◆ You must append the ".value" suffix when accessing the value of a *real variable*, as shown above. This is not the case for local variables.

Finally, you have direct access to all system state, as represented by *object properties*. This uses the same syntax as shown in the "Custom Path" field under a "Property Change" task trigger. In the example shown next to "Property Change Trigger" above, this is:

```
IO.motion1.value
```

Thus, by referring to this property, you could make a wait statement delay further task execution by 5 seconds *only* if the sensor named "motion1" is activated:



2 wait IO.motion1.value ? 5 : 0 seconds

The same rules and capabilities apply in the Trigger Condition field, where you can use the same constants, variables, object properties and operators, plus that you there also have the *trigger* variable when using Property Change or Time of Day triggering (where it contains the new property value, or the current time as a Date object, respectively, as mentioned above).

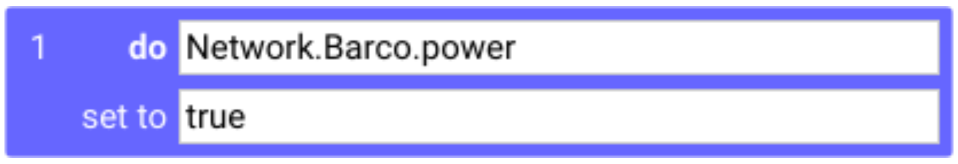
Learn more about expressions in the Blocks wiki: https://pixilab.se/docs/blocks/tasks

do

This is the most commonly used statement. As its name implies, it's the one you use to get things done. It has two forms:

- *Setting a property* on a system object. This uses the same syntax as when assigning a button to a property, but allows you to specify the value using a complex expression, if desired.
- *Calling a function* on a system object. Such *functions* provide additional capabilities not accessible by merely setting a property.

Here's an example of the former, turning on a projector named "Barco", which is managed as a Network device.

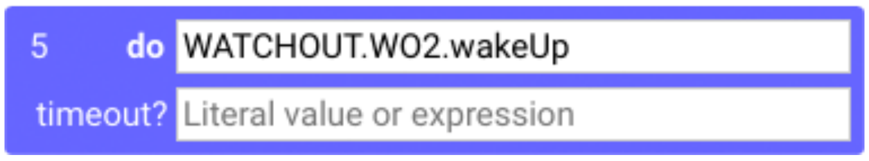


```
1      do  Network.Barco.power
   set to  true
```

Use the Target settings in the settings pane to chose the property to set, just as you do when connecting a button. If you *always* want the projector to be turned on when running this statement, just type *true* into the "set to" field. As this is a literal value, do not type quotes around *true* (see under "What you can Use in Expressions" above).

Calling a Function

Not all control functions are of the simple power or volume type, where a single property value directly represents the feature. For example, you can power up the computers in a WATCHOUT subsystem using the *wakeUp* function. Waking up a cluster is something you may need to do before you can load and play a show, in case its power may have been turned off over night, for example.



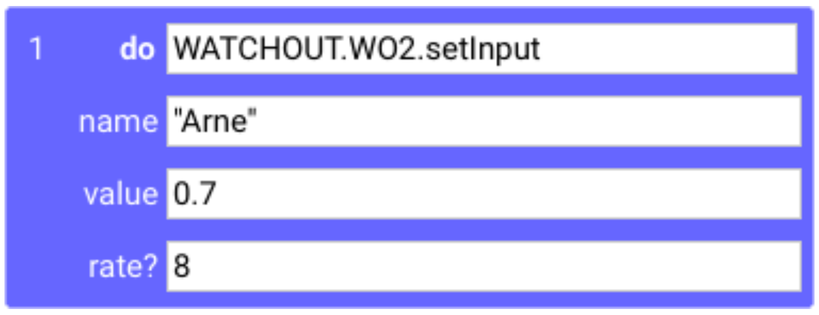
```
5       do  WATCHOUT.WO2.wakeUp
  timeout?  Literal value or expression
```

This function will take some time to complete. You can not proceed with loading a show until all is up and running. The *wakeUp* function handles this automatically, causing the task to pause at this statement until it is all done. To give you some control over how long it may wait, the function takes a *timeout* parameter, allowing you to specify how long you want to wait, at most.

Furthermore, this function parameter is optional, as indicated in its description, and also by the question mark following its name, as seen above. If you don't specify a value, a reasonable default will be applied.

Another example where a function is required is when you need multiple values. For example, when controlling a WATCHOUT *Input* (a WATCHOUT feature often used to control its behavior from outside), you can specify *both* a value *and* a transition rate for that value. While you can bind a slider directly to an input value, this method provides only a single value; namely the value the input will be set to. If you also want to specify the transition rate, you can instead use the *setInput* function. This function takes up to three parameters:

1. The name of the input to set (a string).
2. The value to set it to (a number).
3. An optional transition rate, in milliseconds (a number).



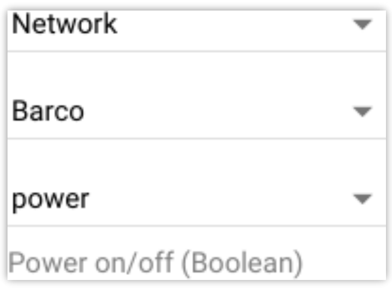
```
1      do  WATCHOUT.WO2.setInput
     name  "Arne"
    value  0.7
    rate?  8
```

If you don't specify a transition rate, it behaves just like directly setting the input's value using a button. But if you set the last parameter to 1000 milliseconds, the Input will instead change gradually, over 1 second, from its current value to the new value.

Using Expressions in Statements

Sometimes, you want a statement to do different things depending on the context. For example, a task that can turn the power *on* or *off* depending on the state of a variable or some other state. If so, use the same *do* statement but replace the literal value *true* with an expression, as shown in the illustration under "Task" at the beginning of this chapter. In that example, the power is set to the value of the TheaterShow realm variable. You can use any valid expression in the "set to" field to determine whether to turn the power on or off.

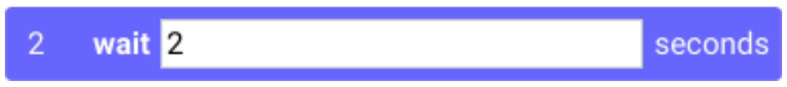
Expression Types and Results



Properties have different types, as indicated at the bottom of the Target settings in the settings pane (see illustration to the right). These types correspond to those described under "Variable Types". The value of an expression used to set the property should generally have the same type as the property being set. For instance, the "power" property of the projector, being of a Boolean type, requires a *true* or *false* value. A volume level, on the other hand, requires a number (usually between 0 and 1). When using an expression, the result is sometimes not quite what you expect. If statements misbehave, you may want to insert a **log** statement with the same expression, just to see what comes out of it.

wait

Delays execution of subsequent task statements by the value of the expression.



You can use an arbitrary expression, yielding a positive numeric value (including fractional parts, such as 0.5 to wait half a second).

log

Outputs the result of the expression to the log window. This window appears below the list of statements when executing a log statement, or when something goes wrong in executing a task. A *log* statement is often used to determine the value of some expression or local variable, and is a valuable troubleshooting aid when tasks don't work as expected.

note

This statement has no effect, and can be used to add notes or comments to a task. For example, explaining what the task does, or some special precautions to keep in mind when later changing the task.

set realm variable

Sets the value of a Realm Variable to the result of the expression or to a literal value. Realm Variables are commonly used to keep track of system state not naturally represented by some already existing system property. They are also useful as a means to transfer information between tasks. Furthermore, a Realm Variable is available as a system property, allowing it to be linked to a button or slider, as well as being used in Task triggers, conditions and expressions.

When accessing a Realm Variable from outside, you do so starting at the Realm level, like this:

```
Realm.Main.variable.TheaterShow.value
```

The Realm Variable in this example is named "TheaterShow", and lives in the Main realm. When referring to the same Realm Variable from within a task in the Main realm, you can simply type

```
TheaterShow.value
```

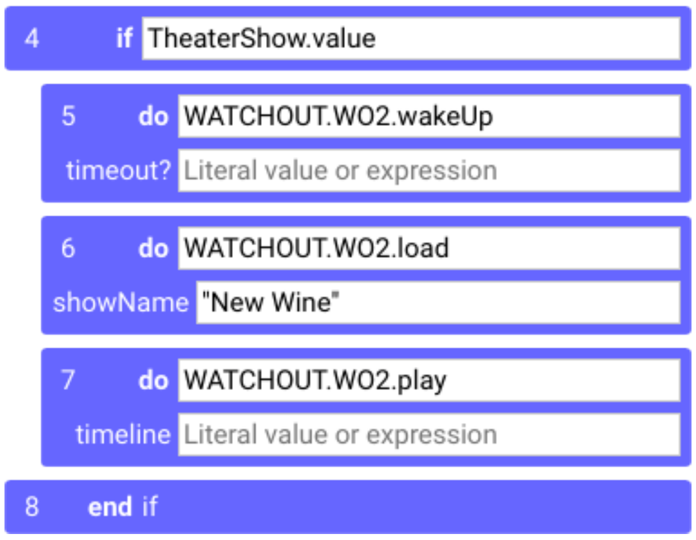
set local variable

A local variable exists only inside the task where it is defined. Use this statement to define a local variable, as well as to change its value. It can not be accessed from outside a task, nor can it be used in task triggers or conditions. A common use for a local variable is to take note of some value at some point in the task, and then read or modify that value at a later point. It's also commonly used with *if* and *while* statements to control their behavior.

A local variable doesn't have a predetermined type, or any other constraints, as Realm Variables do. The variable takes on whatever value you put into it, and may even change type using a new *set local variable* statement setting it to something else. Also, unlike Realm Variables, you can't easily see what value a local variable has. Use a **log** statement to display its current value.

if

An *if* statement allows you to designate a set of statements to be performed *only if* its expression is true. In the example shown below, statement 5, 6 and 7 will only be performed if the TheaterShow.value is true. See the full example at the beginning of this chapter, where this Realm Variable controls whether to turn projectors on or off, and as the task's trigger. The TheaterShow variable is then directly controlled by a button used to start or stop the theater. Only when *true* should the WATCHOUT subsystem be told to wake up, load and play the show, hence the *if* statement.


```
4  if TheaterShow.value
5  do WATCHOUT.WO2.wakeUp
   timeout? Literal value or expression
6  do WATCHOUT.WO2.load
   showName "New Wine"
7  do WATCHOUT.WO2.play
   timeline Literal value or expression
8  end if
```

Statements between if and end if are performed only when the expression in the if statement is true.

If the result of the expression already is a boolean, as in the example above, there's no reason to write

```
TheaterShow.value === true
```

to check it for being true. Instead, just use the boolean value as is:

```
TheaterShow.value
```

Likewise, if you want to perform the statements inside only its value is *false*, you don't need to write

```
TheaterShow.value === false
```

Instead, just prefix the value reference with the *not* operator, like so:

```
!TheaterShow.value
```

Learn more about JavaScript expressions and operators here:

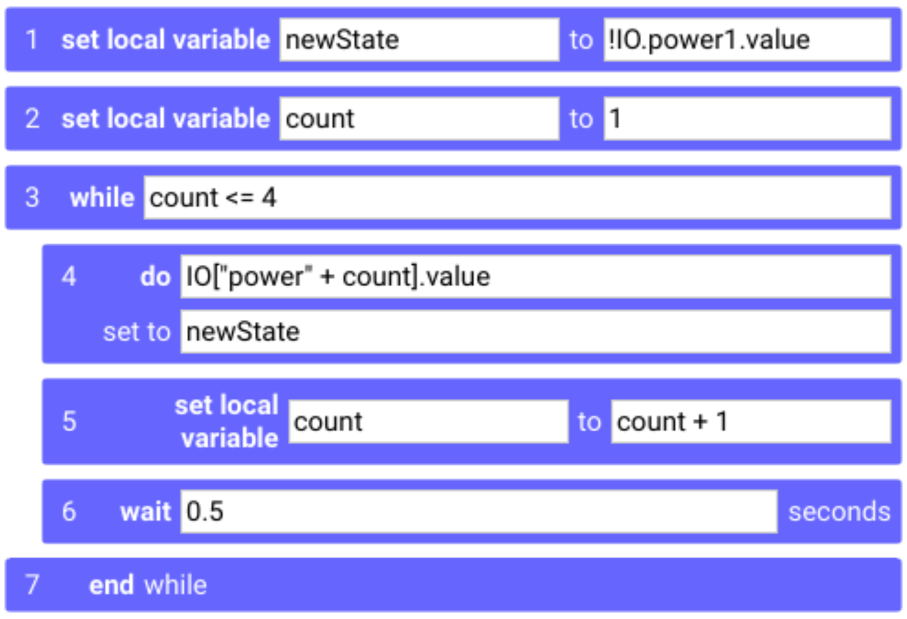
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Expressions_and_Operator

while

The *while* statement is very similar to the *if* statement, but will execute its inner statements *repeatedly* for as long as the condition holds true. Examples of when you may want to repeat some statements are:

- To apply the same set of commands to several targets.
- To repeat some command at a predetermined interval.

While you can, of course, apply the same set of commands to several objects by merely duplicating those statements, changing the target object in each copy, this quickly becomes unwieldy and hard to maintain as the number of targets increases. If the target names contain a consecutive number, such as in the example below, it's easier to use a *while* loop to go through them all. The example shown below deals with turning on a number of power relays in a rack. There are four such relays, each controlling power to a separate part of a building. Rather than turning them all on at once, it may be desired to turn them on in sequence, with a small delay between each relay.



```
1 set local variable newState to !IO.power1.value
2 set local variable count to 1
3 while count <= 4
    4 do IO["power" + count].value
      set to newState
    5 set local variable count to count + 1
    6 wait 0.5 seconds
7 end while
```

The power relays were given consecutive, numbered aliases; power1, power2, power3 and power4. The number at the end of the name can be tacked onto the name using a string concatenation expression, as seen inside the square brackets in the *do* statement. The *count* local variable is initially set to 1 (in statement 2). Concatenating "power" with the *count* value yields "power1", which is the name of the first power relay. The *count* value is then increased by one in statement 5, and the *while* loop repeats the statements as long as the count value is less than or equal to 4 (the number of the last power relay).

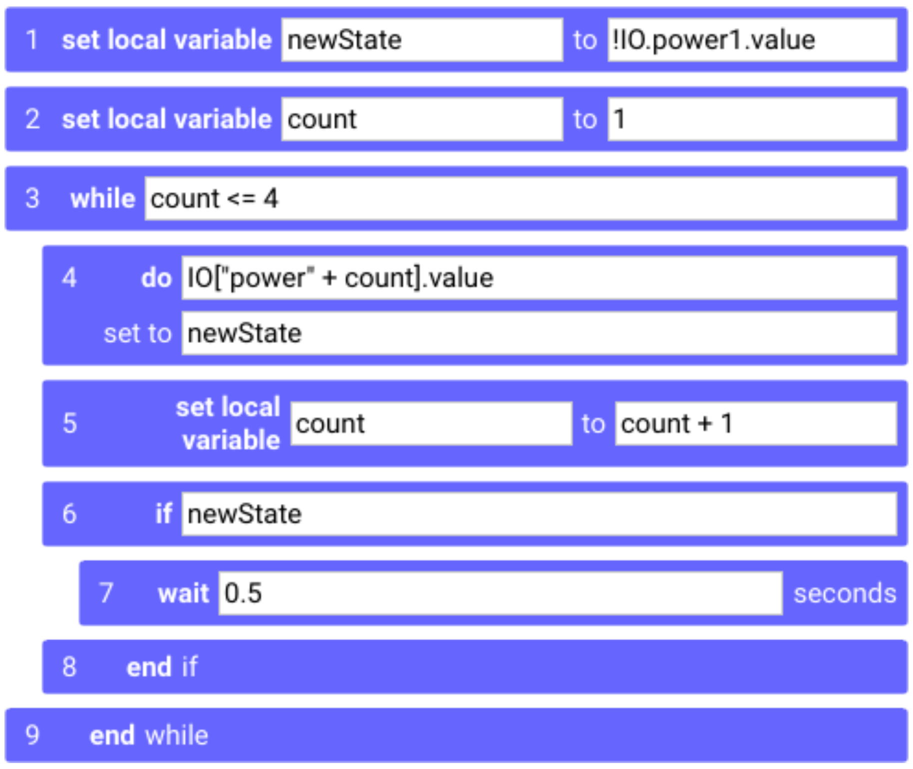
The beauty of this design, compared to merely duplicating the statements inside the *while* loop, is that it is easy to change if the system is expanded to use six power relays instead of four. Just change the test in the while statement to

```
count <= 6
```

and everything else stays the same. Likewise, if you want to increase the delay between each power relay being turned on, you just have to change a single *wait* statement, rather than having to update multiple, duplicated wait statements.

Combining while and if statements

Structural statements, such as **if** and **while**, can be used within each other. In the example discussed above, the delay between switching each power relay is really only needed when turning them *on*. Hence, you can exclude the **wait** statement when turning off by adding an **if** statement, like so:



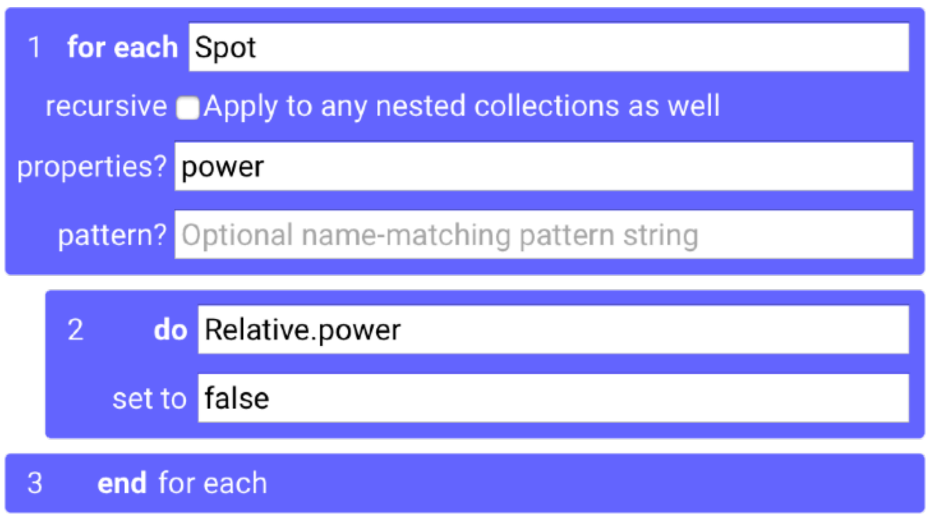
```
1 set local variable newState to !IO.power1.value
2 set local variable count to 1
3 while count <= 4
    4 do IO["power" + count].value
      set to newState
    5 set local variable count to count + 1
    6 if newState
        7 wait 0.5 seconds
    8 end if
9 end while
```

As in the previous example, the local variable *newState* is first set to the desired new state (by picking up the inverse of the old state from the first power relay, using the *not* operator symbolized by the exclamation point). The *newState* variable is then used to set the state of the power relays in statement 4. It is then also used to control the **if** statement, so that the **wait** inside it will only be applied when *newState* is *true* (i.e., the power is turned *on*).

- Even when not using explicit **wait** statements, *all* statements take a fraction of a second to execute. Hence, even after the above change, the relays will not all be turned off exactly simultaneously.

for each

Use **for each** to apply the same set of statements to a collection of Blocks objects, such as Spots, Network Devices or Lighting Fixtures. This is often a convenient alternative to repeating those statements manually.



```
1 for each Spot
  recursive ☐ Apply to any nested collections as well
  properties? power
  pattern? Optional name-matching pattern string
    2 do Relative.power
      set to false
3 end for each
```

1. Select a *Target Collection* using the dropdown menus on the right. This can, for example, be all Spots in the system or a particular Spot Group, all Network or Artnet devices.

2. When choosing a target collection that may contain nested collections – such as *Spot* (as shown in the illustration) or a spot group with nested groups – select *recursive* to apply enclosed statements also to objects inside sub-groups (i.e., inside any nested spot group). If *recursive* is not selected, statement apply only to direct children of the *Target Collection*.

3. To target only objects having a particular property – or set of properties/functions – enter the names of those, comma separated. Only objects having all specified properties/functions will be selected. The illustration above selects only objects having a *power* property. This skips any Visitor Spots or Spot Groups, since those do not have a *power* property.

4. To select objects in the target collection by name, enter a name-matching pattern. This name filter can be combined with the properties filter described in the previous step, in which case only object having those properties *and* matching names will be selected.

The optional name matching described in step 4 above is specified using a regular expression. For instance, to select objects having a name that begins with *Proj* followed by at least one digit, enter a pattern like this:

```
^Proj\d+
```

The regular expression syntax is powerful and flexible, but can be a bit hard to understand. You may want to use an online regex checker to get the syntax right.

For statements inside the *for each*, first select *Relative* at the top level, then select the desired property or function. This is similar to how you access properties of objects inside replicated child blocks.

atomic

Normally, each statement takes 0.05 second to execute (i.e., 20 statements/second). While that's generally fast enough for most control applications, you may occasionally hold statements together as a single temporal unit. For instance, you may want to set a number of lights in a lighting scene simultaneously, rather than 0.05 seconds apart, which may result in a fast ripple-through effect. To do so, wrap those statements to be held together inside atomic/end atomic, and they will all execute together, rather than spread out in time.

Note that within such an atomic section, you may not use any *while* or *wait* statements. Nor will execution pause to wait for any functions to complete (e.g., a *wakeUp* function waiting for the Spot to come online). If desired, you can wrap the *atomic* section inside an *await* section, where further execution will then pause at the *end await* statement, waiting also for any such functions inside the enclosed *atomic* section..

await

The **await** statement allows you to bundle up a number of potentially slow statements, and then wait for them all, rather than waiting for them one at a time.

Some statements take time to complete. The most obvious example is the *wait* statement. But in some cases a *do* statement may take some time to complete. For instance:

- Powering up a Display Spot using the *wakeUp* function when using a PIXILAB Player.
- Loading a show into a WATCHOUT subsystem.

Subsequent statements will not run until the slow statement is done. While having the task wait for each statement to complete before proceeding is usually a desirable behavior, this may get in the way when dealing with multiple, similar functions. For instance, if you want to turn on five displays before starting presentations on these displays. In this case, you probably don't want to wait for each of them in turn before powering up the next one. Instead, you wan to get them all started, then wait for *all* of them to become ready before proceeding.

This can be accomplished using the *await* statement. Simply put the slow statements, to run in parallel, between *await* and *end await*. They will then all be started right away, then the *end await* statement will delay further execution until all of them have completed.

You can tell which functions that may take time to complete by looking at the system object definitions found on PIXILAB's github page. Any function that returns a *Promise* can potentially take a long time to complete. For example, in the WATCHOUT object, you can see that the *wakeUp* function does so. Thus, if called from a task, the task may wait here before moving on. Likewise, a device driver may return a *Promise* in some cases, such as the *powerUp* function of the ChristiePerformance driver also found on github.

14. PROPERTIES AND FUNCTIONS

Properties are exposed by many objects in Blocks. Those properties represent useful system state, such as power being on or off, or a volume level. Such properties can be directly connected to panel controls, such as buttons, sliders, indicators and text. They can also be set or accessed in more advanced control scenarios, such as those described in the previous chapter on Tasks. The set of properties available vary by type of object being managed. For instance, while some Display Spots can be turned on or off by setting their *power* property, the same does not apply to a Visitor spot (for obvious reasons).

- Some properties are provided by device drivers or scripts, or may have been added after the publication of this manual. Hence, always refer to the information shown inside the Blocks editor, as it will be most up-to-date.

Property Values are Bidirectional

A property is represented by a *single value*, such as the power being *true* or *false*, or the volume being at a certain level, say 0.5. This makes a property easy to control – just set its value. This works in both directions; a slider can *control* the volume as well as to *indicate* its current value. If the volume changes, the slider automatically reflects the new level. For example, if a Task uses a *do* statement to change the volume, any slider bound to that volume property will reflect the new value.

This also lets property values act as Task triggers, used in Trigger Conditions and Statement expressions. Thus, any property that represent useful system state, such as the power of something being on or off, can be used in all those places.

Read-only Properties

Some properties merely indicate a state of a device, and can not be set. For instance, the *connected* property of a Spot indicates whether the spot is connected to the server. If not, you won't be able to control the spot, and therefore you may want to know about this. This may be caused by the network cable being unplugged, power being disconnected, or similar. Such read-only properties are shown in *italics* below and on drop-down menus used to select properties. They can not be used as targets of *do* statements, nor can they be controlled by buttons/sliders. But they *can* be used as Task triggers, or in expressions.

Functions

As stated above, a property represents a *single value*. As such, it can be bound directly to a control or indicator. However, some behaviors require multiple values. For instance, to set the color of a lighting fixture, you must provide red, green and blue color components, plus a fade rate (if desired). Such multi-value behaviors can not be managed through a single-value property. In-stead a function must used.

A function is shown in **bold** on drop-down menus. It can only be used from tasks – not be directly bound to a button. To use a function from a button; call the function from a task, then connect the button to the task's *running* property to start it.

Display Spot

A Display Spot manages fixed displays, often connected using wired Ethernet. To access the properties of a Display Spot, chose "Spot" at the top level, then the name of the Spot, or an enclosing Spot Group, finally the name of the Spot itself. Spots may be nested inside Spot Groups to any depth, so you will need to navigate accordingly when specifying the property.

Property	Type	Description
active	boolean	Spot is actively viewed. E.g., using a Locator Block.
block	string	Full name of default block, as 'group/name'.
connected	boolean	True if the player is connected to the server.
pictureSource	string	Name of alternate input source, e.g. 'HDMI', if supported.
playing	boolean	A block is playing (vs. being paused). Indicates that *any* block is playing.

playingBlock	string	Name of currently playing root block. May originate from the assigned block, priorityBlock (see below) or a Schedule Block applied to the spot or an enclosing group.
power	boolean	Set to turn power on/off. Indicates wanted power state. Not supported by all spots. See also the *wakeUp* function below.
priorityBlock	string	Full name of priority block, as 'group/name'. Overrides the default block.
time	time	Time position of playing video/audio, which needs to be under a Synchronizer for this property to work. See also *seek* function below.
volume	number	Audio volume playback level, 0…1. Not supported by all spots. Set Volume Fade Rate under Audio/Video in the spot's settings to make a gradual fade.
keyPress *	boolean	One boolean sub-property per numeric key, for use as general purpose inputs. Available only when "Numeric Keys as Inputs" is enabled in the spot's settings.
keyCode *	string	Name of the most recently pressed key, or empty string if no key pressed. Available only when "Keyboard keyCode Property" is enabled in the spot's settings.
scannerInput *	string	Most recently scanned QR/NFC/RFID tag. Available when "RFID/QR Scanner Input" is enabled or from a QR Scanner block exposed to "scannerInput".
tagSet	string	Comma-separated list of tags applied to this Spot.

◆ **HINT:** Properties marked with an asterisk above can be used locally while offline with *Offline Spot* licensed.

In addition to the properties listed above, a display spot may have any number of parameters. Those can be accessed under

```
Spot.parameter.paramName
```

Where *paramName* is the name of the parameter. Such properties are only available when the spot is connected and *either* has a block with parameters assigned *or* parameters have been added to the spot's settings.

A Display spot also provides the following functions that can be called from tasks and scripts. Parameters within parenthesis are optional, and can be left unspecified.

Function	Parameter	Description
forceTags	tags	Force tags to comma-separated list of tags (plus any in spot's settings).
	(ofSet)	Set of tags to affect (other left alone)
gotoBlock	path	Reveal the specified block path inside the current root block. This uses the same syntax as the "Go to Local Block" Button actions, including numeric and incremental segments. Relative path can not be used. Use the Copy Path button in the breadcrumbs area while editing a nested block to get the path to that block.
	(play)	Play the target block, once found (e.g., a video).
	(seek)	Locate a position in the target block (e.g., a time position in a video).
locateSpot	location	Tell the active Locator to locate specified Spot by its location ID or name (full path).
	(isSpotPath)	If true, location is specified by full spot path (dot separated), else by location ID.
reload	(reloadBrowser)	Reload the current block. Can be used to force block updates programmatically. If *reloadBrowser* is true, the entire web-page will be reloaded, and not just the block.

scrollTo	x	Scroll to specified position (needs a Scroller to work). X position, 0…1 or *undefined*.
	(y)	Vertical position, 0…1
	(time)	Smooth scroll duration, in seconds. Jump-scroll if not specified.
seek	time	Time to seek video/audio to, as a string 'HH:MM:SS.fff". Media must be under a Synchronizer for this property to work. See also *time* property above.
setCustomClasses	(classes)	Class or space-separated classes. Removes all custom classes if empty.
wakeUp	(timeout)	Power up and connect the Spot. Proceed once done.
shutDown		See below. In general, this should be done by setting the *power* property to true.

wakeUp and shutDown
Calling the *wakeUp* function is equivalent to setting the Display Spot's *power* property to true, but will then cause the invoking task to stall until the Spot connects to Blocks, which is occasionally useful if you want to perform some function immediately once the Spot is ready. To power up multiple Spots together, and then perform some function once they are all ready, wrap those wakeUp function calls inside an *await* statement.

The *shutDown* function tells the Display Spot to shut down cleanly, but *without affecting the its 'power' property*. This is occasionally useful prior to turning power off by other means (such as an external mains power switch), allowing the player to shut down cleanly. The rationale for leaving the spot's *power* property unaffected is that explicitly setting *power* to false would then make Blocks insist on keeping the spot's power off indefinitely (until the *power* property is set to tru again). That would interfere with any later attempt to turn power on by the same external means (e.g., mains power switch).

◆ **NOTE:** If the player manages its own power state, you should generally not use this function. Instead, just set the *power* property to false.

Visitor Spot

A Visitor spot is usually connected through cellular or wifi network. You access its properties in the same way as for a Display spot, described above.

Property	Type	Description
block	string	Full name of default block, as 'group/name'.
connected	number	Number of connected visitors.
playingBlock	string	Name of currently playing content block. May originate from schedule.
priorityBlock	string	Full name of priority block, as 'group/name'. Overrides the default block.

◆ **NOTE:** When Visitor Data Collection is being used, and Visitor Identity is enabled for a Visitor Spot, more capabilities become available to User Scripts for managing each individual visitor, matching many of the additional properties, functions and events available for Display Spots. However, these additional capabilities are not accessible from tasks or buttons.

Location

You access the properties of a Location in the same way as for a Display Spot, as described above. The block associated with a Location will appear on Visitor spots when navigating to that Location using a Locator.

Property	Type	Description
block	string	Full name of default block, as 'group/name'.
playingBlock	string	Name of currently playing root block. May originate from schedule. Read only.
priorityBlock	string	Full name of priority block, as 'group/name'. Overrides the default block.

Spot Group

A Spot Group has the same *block* and *priorityBlock* properties as described above. A block assigned to a Spot Group (by drag-and-drop or by setting its *block* or *priorityBlock* properties) applies to Spots inside that group as well, unless overridden. Thus, to assign through the enclosing group, make sure Spots inside the group have no assignment of their own.

Timeline

Timeline properties are accessed by group and name under the top level Timeline object. A Timeline exposes the following properties.

Property	Type	Description
playing	boolean	Timeline is playing. Set to *true* start playback or *false* to pause it.
stopped	boolean	Timeline is stopped (inactive)
time	time	Time position. Use as synchronization source or set to seek.

The *gotoMarker* function can be used to position the timeline at a marker. Use either from the outside (e.g., from a Task or another Timeline) or from within the timeline itself, where this is available under the *Self* top level property path object from a *Do* cue.

Function	Parameter	Description
gotoMarker	cueName	Jump to the Marker cue specified by name, or any marker if the name is blank.
	(play)	Start playback after the jump (else remains paused).
	(searchReverse)	Search only backwards from the current timeline position (else both directions).
	markerTrack	Name of Marker track to search (else searches all marker tracks)

- **HINT:** Use a *Do* cue setting *Self.playing* to false to make the timeline pause at that cue.

Task

Tasks and Realm Variables are accessible under the Realm top-level object, followed by the name of the realm. Then, for a task, select *group*, followed by the group and task name. For a realm variable, select *variable* followed by the variable name.

A Task's *running* property is true while the task is running. Set this property to *true* to start a task from a button or other task. You can also set it to false to terminate the task immediately.

- If the Task has a Trigger Condition, this will be evaluated even when using this method, and the task will *only* start if the condition holds true.

For a Realm Variable, its *value* property contains its current value, which will be of the type specified for the variable. You can set this value from a Button, Slider, Text Input or a Task.

When programming a task within the same realm, use a "set realm variable" statement to set the value. Read its value using just

```
VarName.value
```

where *VarName* is the name of the realm variable. You only need to use the fully qualified name (i.e., beginning with the Realm root level object) when accessing a realm variable from panel controls or tasks in other this realms.

Modbus Channel

Controls devices added under Modbus I/O Modules on the manage page. The only property you can access on a Modbus channel is its value. If the channel is an input, the value is read-only. If the channel is set to "Analog", the value will be a number, otherwise it will be a boolean.

◆ If you give a Modbus channel an *alias name*, you can then also access that channel with that alias under the top-level IO namespace. This lets you assign a name that describes the function in a way that is independent of its module name, and that can remain the same even if the function is moved to another Modbus module (e.g., for wiring reasons).

Art-Net/DMX

To access a lighting fixture configured under "Art-Net/DMX-512 Lighting", first select Artnet then the name of the fixture, followed by the desired property or function. Most channels have a *value* property, which in many cases is normalized to the range 0…1, regardless of its actual resolution. The range is generally indicated below the selected property.

If you set the property's value, the Default Fade Rate specified for the fixture will be applied. If you instead chose any *fadeTo* function, you can specify both the target value and the fade rate.

For some lights, with common functions such as RGB color channels, or pan/tilt functionality, an additional group of functions named *multiFunc* may be provided, allowing you to set multiple values with a single task statement.

Some properties may have a discrete (enumerated) value, which is then specified by a string. For properties that have both a discreet state and an associated numeric value, first select the discrete state then (using a second **do** statement) set the associated value.

Art-Net recordings, if any, appear by their respective name under a Recording subheading under the Artnet top level item. A recording has the following features from a task/script's point of view:

- A **playing** property; set while the recording plays. To play the recording in free-running mode, set this property to true. While in synchronized playback mode (see below) this property is read-only, and will be true while the recording is playing.
- A **time** property, indicating the time position during playback, usable for synchronization purposes.
- A **startSync** function, activating synchronized playback. This function takes two parameters.
  – The first parameter specifies the path to the property (of *time* type, known as TimeFlow in scripts) acting as the synchronization time source. If set to empty string or null, this is equivalent to calling *stopSync*.
  – The second (optional) parameter specifies an offset, in seconds, added to the time received from the time source. This offset may be negative and accepts fractions (e.g., -1.7).
- A **stopSync** function, stopping and terminating any synchronized playback.

WATCHOUT

Controls a WATCHOUT subsystem added under WATCHOUT Clusters on the Manage page.

Property	Type	Description
auxTimeline	timeline	Auxiliary timelines (if any).
input	number	Current value of the input (as set by Blocks)
connected	boolean	True if the cluster is connected, false if cluster is offline.
layerConditions	number	Combined numeric value of enabled layer conditions. WATCHOUT version 6 only.
playing	boolean	True if the main timeline is playing.
ready	boolean	True if the cluster is fully operational, false if cluster is “busy” (e.g., downloading files).
showName	string	Name of the currently loaded show. Name only for display PC, or full path (including “.watch” extension) for production PC. WATCHOUT version 6 only.
standBy	boolean	True while cluster is in stand-by mode. WATCHOUT version 6 only.
time	time	The current time position of the main timeline. Specified in milliseconds from tasks.

When choosing auxTimeline, you then need to choose or enter the name of the auxiliary timeline, which then provides access to the following sub-properties:

Property	Type	Description
playing	boolean	True if the auxiliary timeline is playing, false if paused/stopped.
stopped	boolean	True if the auxiliary timeline is stopped. False if playing.
time	time	The current time position of the auxiliary timeline. Specified in milliseconds from Tasks.

In addition to the above properties, WATCHOUT clusters also provide the following functions that can be called from tasks:

Function	Parameter	Description
connect		Attempt to connect to the cluster. Proceeds once finished.
disconnect		Disconnect from cluster.
gotoControlCue	name	Name of control cue to jump to.
	reverseOnly	Search only backwards in time for the specified control cue.
	timeline	Name of timeline to control (main timeline if not specified).
gotoTime	time	Time to go to (specified in milliseconds).
	timeline	Name of timeline to control (main timeline if not specified).
load	showName	Load a show, deferring further task execution until done. Specify full path name if controlling production PC, else just show name. Will always load, even if the same show is already loaded.
pause	(timeline)	Pause timeline. Name of timeline, empty for Main.
play	(timeline)	Play timeline. Parameter is name of timeline, empty for Main

reset		Stop all timelines.
setInput	name	Set named Input value.
	value	Value to set it to
	(rate)	Transition rate, in mS.
setStandBy	standby	Set cluster Stand-By state. WATCHOUT version 6 only.
	(rate)	Rate of transition, in milliseconds.
stop	(timeline)	Stop timeline. Name of timeline, empty for Main.
sleep		Disconnect, shut down and power off all display PCs in the cluster. Version 6 only
wakeUp	(timeout)	Power up and connect all display PCs in the cluster. Proceeds once done. Version 6 only

Schedule

Each Block Scheduler exposes the *group/name* of its current block as a property under the Schedule top level object. One way of using this property is as the target of a Reference Block, allowing you to schedule part of a Composition, which then becomes an independently scheduled "zone".

Network

Controls devices added under Network Devices on the manage page.

Property	Type	Description
connected	boolean	True if a connection to the device, MQTT broker or Serial port was established.
name	string	The name given to the network device
fullName	string	Full path to the device (i.e., its name prefixed by "Network.".
driverName	string	The name of any associated driver.
deviceType	string	One of "NetworkTCP", "NetworkUDP", "MQTT" or "Serial"

When used with a device driver, the driver typically exposes a number of higher level properties (such as power on/off or input selection when controlling a video projector). It may or may not expose the properties and functions listed here.

Function	Parameter	Description
enableWakeOnLAN		Prepares this device to act on wakeOnLAN command (remembers its MAC address).
sendBytes	toSend	Send array of raw data bytes.
sendText	toSend	The text to send to the remote device.
	(eoln)	End of line sequence. Defaults to carriage return.
wakeOnLAN	(macAddress)	Send wake-on-LAN to this device or the specified MAC address (optional, expressed as a colon separated string of HEX bytes, or an array of 6 numbers).

15. AUTHORIZATION AND ROLES

Blocks defines a fixed set of roles that can be assigned to users. This is done under Manage > Users (see "Users"). A user is always requested to log in when using the Blocks editor. When accessing a Spot, log-in is optional, and is generally not used (see "Role Required to Use Spot").

The following user roles are available:

Role	Description
Admin	Full control over to all aspects of Blocks, including users and their roles.
Manager	Full control over to all aspects of Blocks, except users and their roles
Creator	Can create and edit content of all kinds.
Editor	Can edit existing content.
Contributor	Can replace content in top level text and media blocks.
Staff	A logged in user, that can't edit anything (can be used for control access).

Each role constitutes a proper subset of the role above. Likewise, each higher role has all the capabilities of the lower roles, plus its own. The same holds true when using these roles to configure access restrictions, as described below. This is also indicated by the following table, describing the Blocks editor permissions of each role.

Function	Admin	Manager	Creator	Editor	Contributor
User add/delete/configure	✔				
Managed objects add/configure	✔	✔			
Managed objects delete/rename	✔	✔			
Spot add/configure/reassign	✔	✔			
Spot delete/rename	✔	✔			
Schedule add	✔	✔	✔		
Block duplicate/edit	✔	✔	✔	✔	
Block add, rename, edit	✔	✔	✔		
Make or change block template or protection	✔	✔	✔		
Block delete	✔	✔	✔		
Schedule edit	✔	✔	✔	✔	
Schedule delete/rename	✔	✔			
Edit/replace root media/text block content	✔	✔	✔	✔	✔
Task/Group/Variable add/configure/edit	✔	✔			
Task/Group/Variable delete/rename	✔	✔			

Configurable Access Restrictions

In addition to the fixed editor restrictions listed above, it's also possible to configure object-specific restrictions for various functions.

Function	Allowed by
View a Display or Visitor spot	Spot's "Role Required to Use" setting.
See a Block in the Editor	Block's or Group's "Role Required to See" setting.
Assign (drag) Block or Schedule to a Spot	Spot's or enclosing Group's "Role Required to Control" setting.

As discussed above, roles constitute proper subsets. Hence, if you select the Creator role as the restriction for access, you'll implicitly also grant access to users that have the Manager and Admin role.

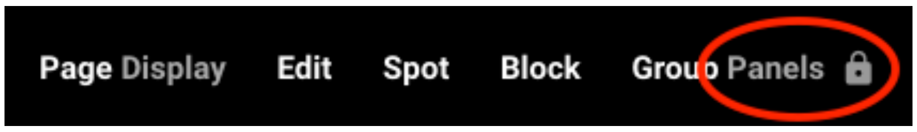
Role Required to Use a Spot

You can restrict who can access a Display or Visitor spot based on role. This is done using the "Role Required to Use Spot" setting found under the Advanced tab of the spot's settings.

If you select any role except *None* here, the spot will require you to log in before it will display its content. Thus, this setting is only useful with display devices that have a keyboard for entering credentials. This option is intended for Display or Visitor spots used as control panels for restricted system objects. A button or other control attached to a restricted system objects has no effect unless the spot has already been logged in by a user with sufficient role, as governed by the "Role Required to Control" settings of each system object (see below).

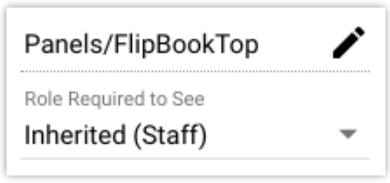
Role Required to see Block in the Editor

You can also restrict who can see certain blocks in the Editor. By hiding blocks from some users, you reduce the complexity and chance for errors, showing only the relevant blocks as determined by the role of the logged-in user. You can specify the role required to see blocks either at the group level or for each individual block. Hence, if blocks that are "off limits" for some roles are already grouped together, you can simply set the restriction at the group. This is done by first selecting the group using the menu bar at the top of the Display page, and then clicking the padlock symbol next to the group name.



This brings up a dialog box, allowing you to specify the minimum role required to see blocks in this group in the editor. Once set, such blocks will be hidden from users with a lower role.

Alternatively, you can set the minimum role required to see the block in the block itself. This setting is available only at the topmost level of the block. By default, blocks inherit the role specified at the group level, as indicated by the "Inherited" setting shown in the illustration to the right. However, you can use the block level setting to override the group level, making it more restrictive or more permissive.


Panels/FlipBookTop

Role Required to See
Inherited (Staff)

Role Required to Control

Blocks provides granular control over who can control what, based on the roles assigned to users. Such control restrictions are specified on a per-object basis, and are represented by padlock icons throughout Blocks. These padlocks govern external control originating from buttons and other controls appearing on spots. Initially, the role required to control all objects is set to *None*, meaning that no specific role is required to control objects. This permissive setting matches the behavior of previous versions of Blocks, and guarantees that control functions continue to work in the same manner as before.

These control restrictions use the roles listed above, plus a pseudo-role called *Identified Spot*. This pseudo-role does not require log-in, and is given to all Display and Visitor spots in the system except the default Visitor spot as specified by the serverRoot-Redirect configuration setting. This setting provides some small amount of security, since it means that a visitors using the default Visitor spot name can not control such objects. Only spots that identify themselves by an ID known to the system, or that use an explicit Visitor spot URL, qualify as Identified Spots. Hence, this is a useful default setting for all objects not intended to be controlled by random visitors (or hackers).

All roles except *None* and *Identified Spot* require a logged in user to control the object. Furthermore, the logged in user must have the specified role or higher to control the object. The log-in behavior must be triggered by the "Role Required to Use a Spot" setting discussed above. If a user with the required role isn't logged in, the button or other control won't work, and an error message to that effect will be logged.

While you of course wouldn't provide "dangerous" control functions on publicly available visitor spots, this explicit restriction affords additional protection against malicious attempts to control system functions. It acts as an internal fence, verifying all attempts coming from the outside, making sure that they obey any access rules defined for each object.

◆ This "Role Required to Control" restriction applies *only* to control attempts originating from outside (e.g., from spots). Internal control within Blocks itself, such as tasks and scripts, is not subject to these restrictions.

The shape of the padlock symbol indicates what kind of restriction that is applied at this level.

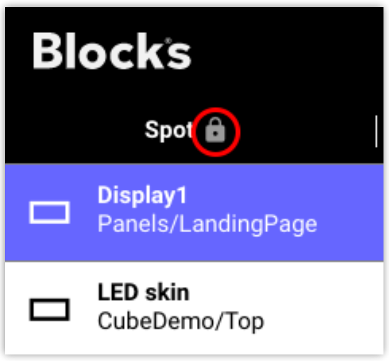
🔓 No control restriction applies.

🔒 Control restriction applies.

🔒 A stricter control restriction applies compared to the parent object or group.

🔒 A less strict control restriction applies compared to the parent object or group.

Spots
For spots, the "Role Required to Control" governs all properties, such as *block*, *priorityBlock* and *power*. The role can be set under ADVANCED in the spot's settings. If not explicitly overridden, it is inherited from any enclosing Spot Group (as specified in the group's settings), or – at the top level – from the global spot "Role Required to Control" setting accessed through the padlock symbol at the top of the Spot list.

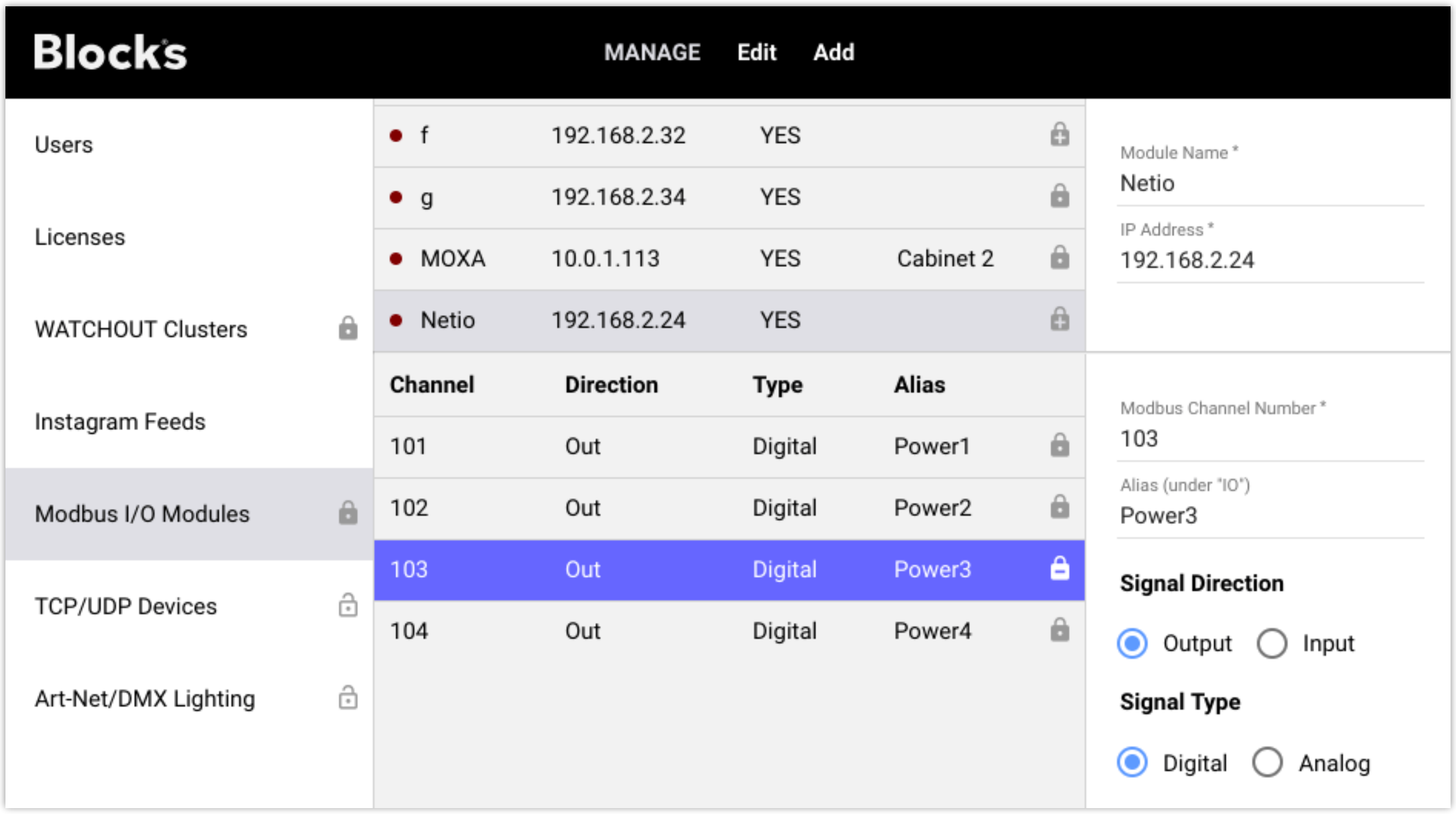


Managed Objects
All controllable objects found on the Manage page have control restriction settings, as indicated by the padlocks found in various places under Manage. The padlock settings shown in the list of managed objects specify the default restriction each category (i.e., WATCHOUT Clusters, Modbus I/O modules, TCP/UDP Devices and Art-Net/DMX Lighting).

Selecting one of these categories shows its objects, where each of those objects can override the setting inherited from the category level. Finally, for some objects (such as Modbus outputs), you can override this again for each individual output.

In the illustration shown below, restrictions are applied to WATCHOUT Clusters and Modbus I/O Modules at the category level. The "Netio" Modbus module has a more restrictive role requirement, as indicated by the plus sign in its padlock symbol. All its output channels have the same requirements except channel 103 which is less restrictive.

To see the role requirement set at each level, and to change this setting, click the padlock symbol at the desired level.



● f	192.168.2.32	YES	
● g	192.168.2.34	YES	
● MOXA	10.0.1.113	YES	Cabinet 2
● Netio	192.168.2.24	YES	

Channel	Direction	Type	Alias
101	Out	Digital	Power1
102	Out	Digital	Power2
103	Out	Digital	Power3
104	Out	Digital	Power4

Module Name *
Netio

IP Address *
192.168.2.24

Modbus Channel Number *
103

Alias (under "IO")
Power3

Signal Direction

[✓ Output] Input

Signal Type

[✓ Digital] Analog

Tasks and Realm Variables

A similar hierarchy of control restrictions applies to tasks and realm variables. The top level restriction is here specified at the Realm level, as set by the padlock next to the realm name at the top of the window. This setting determines the restriction applied to its realm variables and task groups, unless overridden. Any setting applied at the group level is inherited by tasks in that group, unless overridden by a particular task.

For tasks, this restriction controls who can start the task by setting its *running* property from a button. It doesn't affect normal triggering of tasks, as specified by each task's *Trigger* setting.

Likewise, for Realm variables, it governs who can set the variable's value using a button, slider or other control. It does not affect the ability to set realm variables internally (e.g., from task statements).

◆ Since tasks are rarely intended to be started directly by anonymous visitors, it's generally a good idea to set the Realm level role requirement to a stricter setting than *None*. This can then be relaxed for individual tasks or variables, if required.

A. SERVER FILE STRUCTURE

The Blocks server stores its data as well as your media files and other settings in a folder structure. While you don't need to know what this looks like to use Blocks, it can occasionally be useful to know where things are, for example if you want to move things from one Blocks server to another.

By default, Blocks will keep its data in a directory named "PIXILAB-Blocks-root", stored in the home directory of the user account under which the server is run.

PIXILAB-Blocks-root	Root directory of all Blocks data, located in your user/home directory.
config.yml	Server configuration file (see "Server Configuration Options").
logs	Detailed log files, sometimes useful when troubleshooting server errors.
model	Settings from the Manage and Task pages live in here, as well as Spots and Schedules.
Core	Internal server data, uniquely identifying this Blocks server.
DmxRecordings	Settings and data related to Art-Net data recordings.
ModbusModules	Contains one file per Modbus module added on the Manage page.
NetworkDevices	Contains one file per Network module added on the Manage page
Schedules	Contains all your Schedule blocks
Main	Directory containing the "Main" group (one directory per group)
Entrance.json	File describing the "Entrance" schedule of the "Main" group.
Spot	File describing the configuration of all Spots.
Task	Directory containing all your Realms (one file per realm).
Users	File with information about all Users added through the Manage page.
Licenses-XXXX	Additional licensing-related information. Do not remove or share.
WOClusters	Contains one file per WATCHOUT subsystem added on the Manage page.
public	Directory containing publicly (http) accessible files.
block	All blocks you create (except Schedules) live in here.
Main	A block group named "Main".
Saltkrakan	The folder of block named "Saltkrakan".
media	The media files used in this block live in this directory.
Spec.pixi	Data describing the block's structure.
script	
driver	Driver files for network devices.
user	User scripts.
feed	Database feed scripts.

Transferring Server Configuration

You can transfer entire block groups as well as individual blocks by copying the block or its entire enclosing group folder from one Blocks server to another. You don't need to restart the server after adding new blocks to it in this way. Just reload the Display page in the editor to make the new blocks appear.

Likewise, you can transfer schedules, network devices, modbus modules, tasks and device drivers by copying those directories or files from one Blocks server to another. Here, however, you should quit the server before adding those files, and then restart the server.

Backing up the Server

Finally, to make a backup of your entire Blocks server configuration, just copy the "PIXILAB-Blocks-root" directory to a safe place. To later restore this backup, quit the server and replace the "PIXILAB-Blocks-root" directory with the previously saved one. More on backups here.

B. SCRIPTING AND DEVICE DRIVERS

In addition to the task-based programming described earlier in this manual, Blocks also supports *advanced scripting*, using a programming language called TypeScript. Scripting can be used to handle very complex scenarios, such as game engines, where more traditional programming is often called for.

Learn more about the requirements and use cases for scripting in the PIXILAB Wiki.

Developing Device Drivers

The TypeScript programming language is also used for developing custom *device drivers*, discussed earlier in this manual. Such development requires similar programming skills, and is also described in the PIXILAB Wiki.