Telit deviceWISE

deviceWISE VIEW Cookbook and User Guide

2022-01-17

TELIT
TECHNICAL
DOCUMENTATION

01.2022

Mod. 2022-01 Rev.6

TABLE OF CONTENTS

SPECIFICATIONS ARE SUBJECT TO CHANGE WITHOUT NOTICE

NOTICE

While reasonable efforts have been made to assure the accuracy of this document, Telit assumes no liability resulting from any inaccuracies or omissions in this document, or from use of the information obtained herein. The information in this document has been carefully checked and is believed to be reliable. However, no responsibility is assumed for inaccuracies or omissions. Telit reserves the right to make changes to any products described herein and reserves the right to revise this document and to make changes from time to time in content hereof with no obligation to notify any person of revisions or changes. Telit does not assume any liability arising out of the application or use of any product, software, or circuit described herein; neither does it convey license under its patent rights or the rights of others.

It is possible that this publication may contain references to, or information about Telit products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that Telit intends to announce such Telit products, programming, or services in your country.

COPYRIGHTS

This instruction manual and the Telit products described in this instruction manual may be, include or describe copyrighted Telit material, such as computer programs stored in semiconductor memories or other media. Laws in the Italy and other countries preserve for Telit and its licensors certain exclusive rights for copyrighted material, including the exclusive right to copy, reproduce in any form, distribute, and make derivative works of the copyrighted material. Accordingly, any copyrighted material of Telit and its licensors contained herein or in the Telit products described in this instruction manual may not be copied, reproduced, distributed, merged, or modified in any manner without the express written permission of Telit. Furthermore, the purchase of Telit products shall not be deemed to grant either directly or by implication, estoppel, or otherwise, any license under the copyrights, patents, or patent applications of Telit, as arises by operation of law in the sale of a product.

COMPUTER SOFTWARE COPYRIGHTS

The Telit and 3rd Party supplied Software (SW) products described in this instruction manual may include copyrighted Telit and other 3rd Party supplied computer programs stored in semiconductor memories or other media. Laws in the Italy and other countries preserve for Telit and other 3rd Party supplied SW certain exclusive rights for copyrighted computer programs, including the exclusive right to copy or reproduce in any form the copyrighted computer program. Accordingly, any copyrighted Telit or other 3rd Party supplied SW computer programs contained in the Telit products described in this instruction manual may not be copied (reverse engineered) or reproduced in any manner without the express written permission of Telit or the 3rd Party SW supplier. Furthermore, the purchase of Telit products shall not be deemed to grant either directly or by implication, estoppel, or otherwise, any license under the copyrights, patents or patent applications of Telit or other 3rd Party supplied SW, except for the normal non-exclusive, royalty free license to use that arises by operation of law in the sale of a product.

USAGE AND DISCLOSURE RESTRICTIONS

I. License Agreements

The software described in this document is the property of Telit and its licensors. It is furnished by express license agreement only and may be used only in accordance with the terms of such an agreement.

II. Copyrighted Materials

Software and documentation are copyrighted materials. Making unauthorized copies is prohibited by law. No part of the software or documentation may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form or by any means, without prior written permission of Telit

III. High Risk Materials

Components, units, or third-party products used in the product described herein are NOT fault-tolerant and are NOT designed, manufactured, or intended for use as on-line control equipment in the following hazardous environments requiring fail-safe controls: the operation of Nuclear Facilities, Aircraft Navigation or Aircraft Communication Systems, Air Traffic Control, Life Support, or Weapons Systems (High Risk Activities"). Telit and its supplier(s) specifically disclaim any expressed or implied warranty of fitness for such High Risk Activities.

IV. Trademarks

TELIT and the Stylized T Logo are registered in Trademark Office. All other product or service names are the property of their respective owners.

V. Third Party Rights

The software may include Third Party Right software. In this case you agree to comply with all terms and conditions imposed on you in respect of such separate software. In addition to Third Party Terms, the disclaimer of warranty and limitation of liability provisions in this License shall apply to the Third Party Right software.

TELIT HEREBY DISCLAIMS ANY AND ALL WARRANTIES EXPRESS OR IMPLIED FROM ANY THIRD PARTIES REGARDING ANY SEPARATE FILES, ANY THIRD PARTY MATERIALS INCLUDED IN THE SOFTWARE, ANY THIRD PARTY MATERIALS FROM WHICH THE SOFTWARE IS DERIVED (COLLECTIVELY “OTHER CODE”), AND THE USE OF ANY OR ALL THE OTHER CODE IN CONNECTION WITH THE SOFTWARE, INCLUDING (WITHOUT LIMITATION) ANY WARRANTIES OF SATISFACTORY QUALITY OR FITNESS FOR A PARTICULAR PURPOSE.

NO THIRD PARTY LICENSORS OF OTHER CODE SHALL HAVE ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND WHETHER MADE UNDER CONTRACT, TORT OR OTHER LEGAL THEORY, ARISING IN ANY WAY OUT OF THE USE OR DISTRIBUTION OF THE OTHER CODE OR THE EXERCISE OF ANY RIGHTS GRANTED UNDER EITHER OR BOTH THIS LICENSE AND THE LEGAL TERMS APPLICABLE TO ANY SEPARATE FILES, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

1. INTRODUCTION

1.1. What is deviceWISE VIEW?

deviceWISE VIEW provides a user-friendly way of collecting and visualizing real-time machine data using only a few clicks. Transform your data and calculate your KPIs with the deviceWISE Gateway visual edge logic engine. Create custom dashboards and HMIs that can be displayed on-premises or in the cloud easily using drag-and-drop widgets. Visualizing your factory data can help you make smarter business decisions by noticing patterns and trends, analyzing data, and interpreting the data you receive.

1.2. Cookbook Requirements

The following items are required for successful use of deviceWISE VIEW following the steps provided in this guide.

- deviceWISE Gateway
- deviceWISE Workbench
- deviceWISE VIEW Licenses

For instructions on how to install the deviceWISE Workbench and Gateway, visit:

http://docs.devicewise.com

1.3. Audience

This document is intended for deviceWISE VIEW users who desire to build their first display and understand the various widgets it has to offer. This guide also provides more advanced or ambitious users with numerous tricks and tools involving subtriggers, database tables, animation, and creating range-based data trends, all while utilizing the extensive in-depth customization options included with deviceWISE VIEW.

1.4. Contact Information, Support

For general contact, technical support services, technical questions, and report documentation errors, contact Telit Technical Support at:

- TS-EMEA@telit.com
- TS-AMERICAS@telit.com
- TS-APAC@telit.com

Alternatively, use:
http://www.telit.com/support

For detailed information about where you can buy Telit modules or for recommendations on accessories and components visit:
http://www.telit.com

Our aim is to make this guide as helpful as possible. Keep us informed of your comments and suggestions for improvements.

Telit appreciates feedback from the users of our information.

1.5. Text Conventions

Danger – This information MUST be followed or catastrophic equipment failure or bodily injury may occur.

Caution or Warning – Alerts the user to important points about working with the solution, if these points are not followed, the solution may fail or malfunction.

Tip or Information – Provides advice and suggestions that may be useful when working with the software.

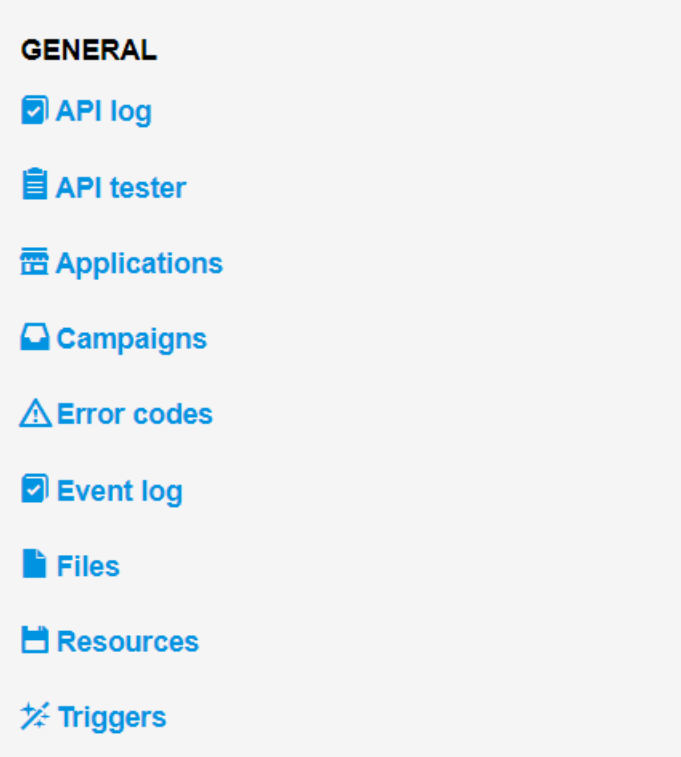
All dates are in ISO 8601 format, i.e. YYYY-MM-DD

2. PREPARING FOR DEVICEWISE VIEW

Before being able to access deviceWISE VIEW, you must first download the required package from the Telit IoT Portal. While VIEW works on a variety of supported gateways and operating systems, a Windows 64-bit computer will be used for the sake of this guide.

2.1. Downloading the Required Package

2.1.1. To install the deviceWISE VIEW package, log into the Telit IoT Portal and navigate to the 'Developer' tab. Select 'Resources' from the left side bar.



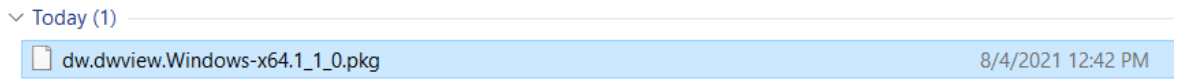
2.1.2. Navigate to the package by opening the build version your gateway is running -> selecting your gateway or operating system -> Packages. Within the 'Pilot_Ready/' folder, download the package containing 'dw.dwview'.

This guide assumes the reader has the deviceWISE Asset or Enterprise Gateway already installed on their system and it has been added as a node within the deviceWISE Workbench. This guide utilizes deviceWISE VIEW version 1.3.0 and may contain features that are unavailable in previous releases.

For more information regarding the installation of the deviceWISE Gateway, the deviceWISE Workbench, or for further reference content and up-to-date information regarding deviceWISE VIEW, please visit:

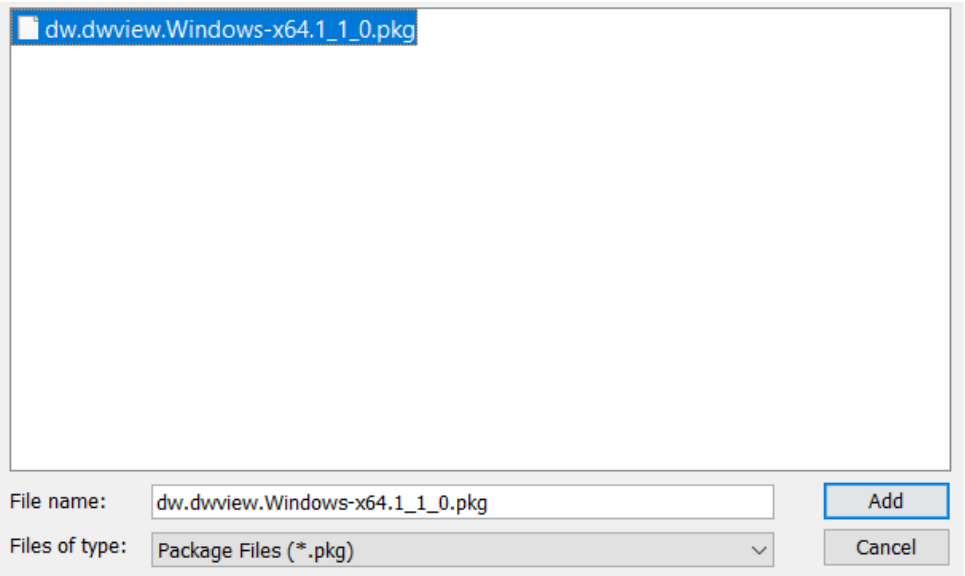
http://docs.devicewise.com

For Windows 64-bit, this package will be called ‘dw.dwview.Windows-x64.<VERSION>.pkg’.

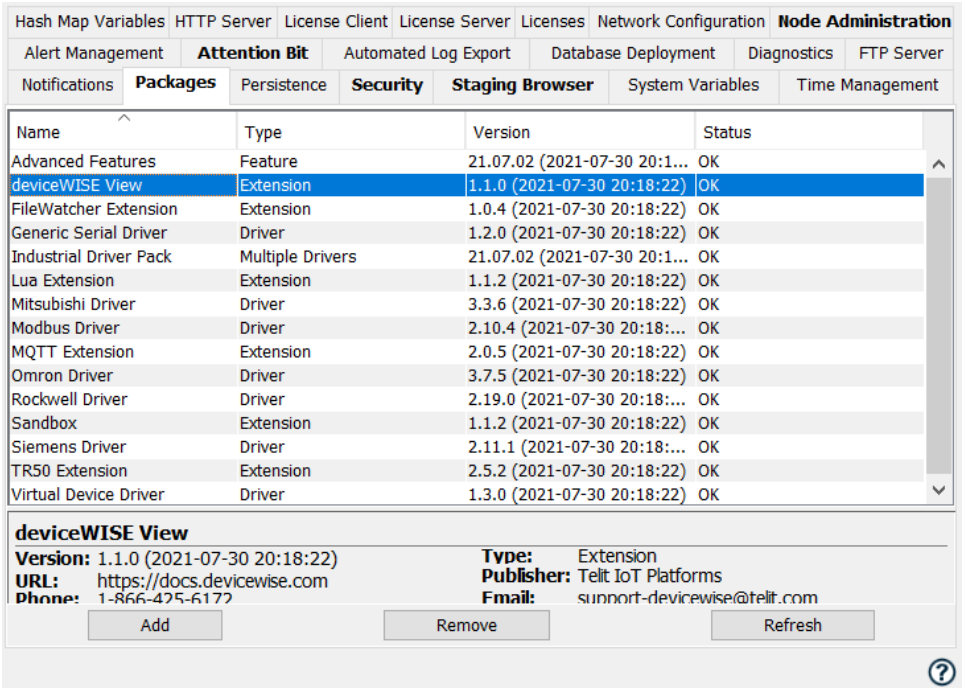


2.2. Adding a Package

2.2.1. In the deviceWISE Workbench, select the ‘Administration’ drop-down from the left side panel of your node and navigate to the ‘Packages’ tab. To add the deviceWISE VIEW package, click ‘Add’ near the bottom of the window and locate where you saved the package on your computer. Select the package and click ‘Add’. The node will restart.



2.2.2. After the node has finishing restarting, navigate back to the ‘Packages’ tab within the Workbench and ensure that ‘deviceWISE View’ listed with an ‘OK’ status.

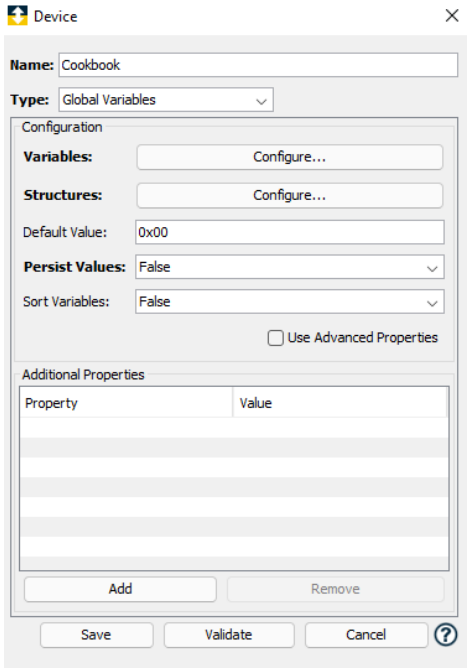


2.3. Creating Variables

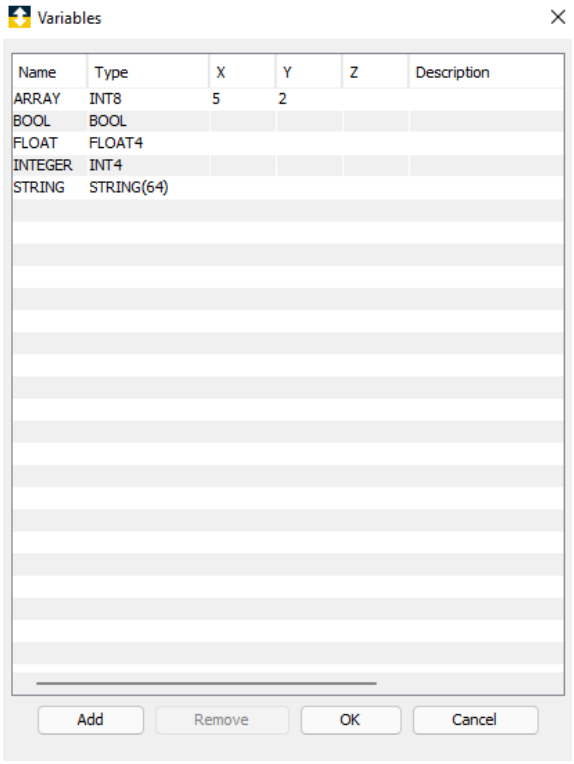
In the deviceWISE Workbench, we will be creating several global variables for deviceWISE VIEW to later interact with. The variable types we will be using are BOOL, INT4, INT8, FLOAT4, and STRING.

2.3.1.

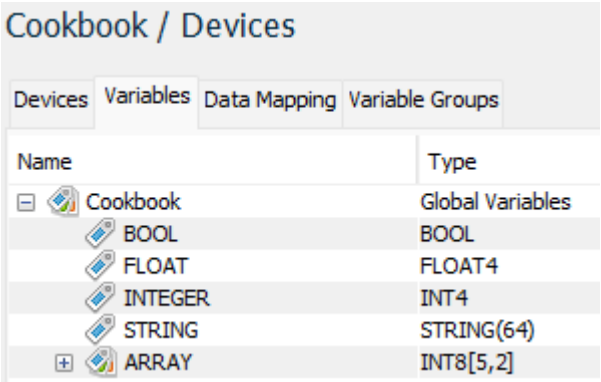
Navigate to the ‘Devices’ panel on the left side bar of the Workbench. To create a set of global variables, we must create a new device by clicking ‘New’ near the bottom of the window. Name the device “Cookbook” and choose ‘Global Variables’ as the ‘Type’.



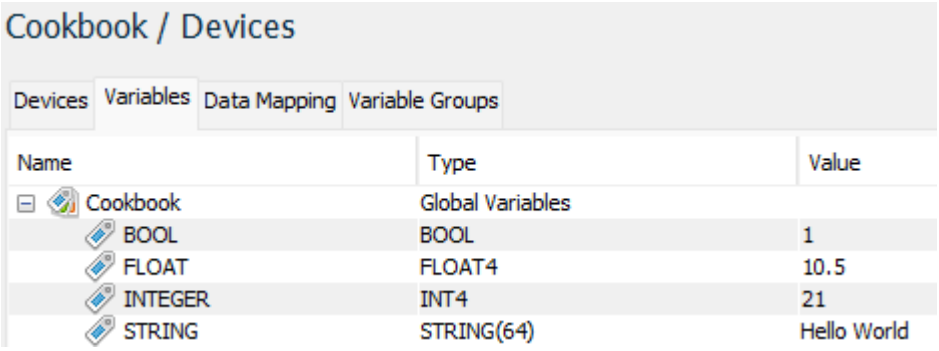
2.3.2. Click ‘Configure…’ next to ‘Variables’ to begin configuring variables. Define five (5) variables as shown below.



2.3.3. Click ‘OK’, then ‘Validate’ and ‘Save’ the device. To access these variables using deviceWISE VIEW, we must start it by right clicking on the newly created ‘Cookbook’ device and clicking ‘Start’. Once started, verify that the variables can be seen under the ‘Variables’ tab at the top of the window.



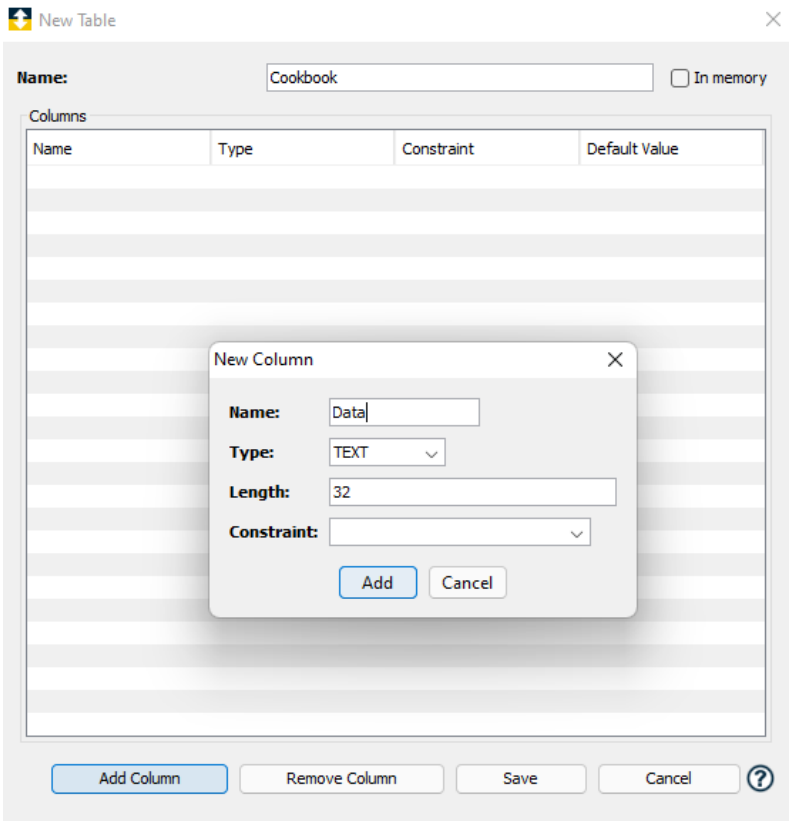
2.3.4. Write to the first four (4) variables by right clicking them and selecting ‘Write’. deviceWISE VIEW will be able to read these values later. Example values are shown below.



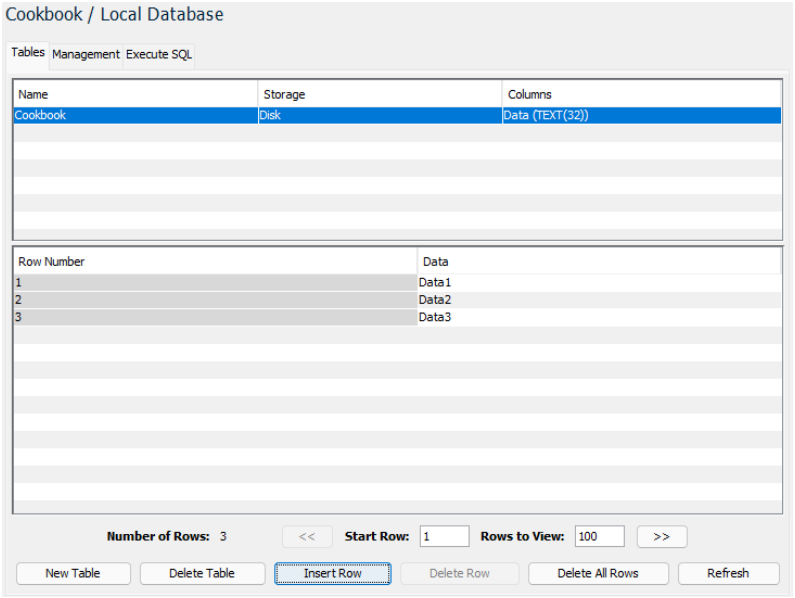
2.4. Creating a Local Database Table

deviceWISE VIEW also has the ability to interact with deviceWISE local database tables. We will be creating a database table now to work with later in the guide.

2.4.1. In the Workbench, navigate to 'Local Database' from the left panel and create a 'New Table'. Name the table "Cookbook" and click 'Add Column'. Name the new column "Data" and then 'Add' and 'Save' the database table.



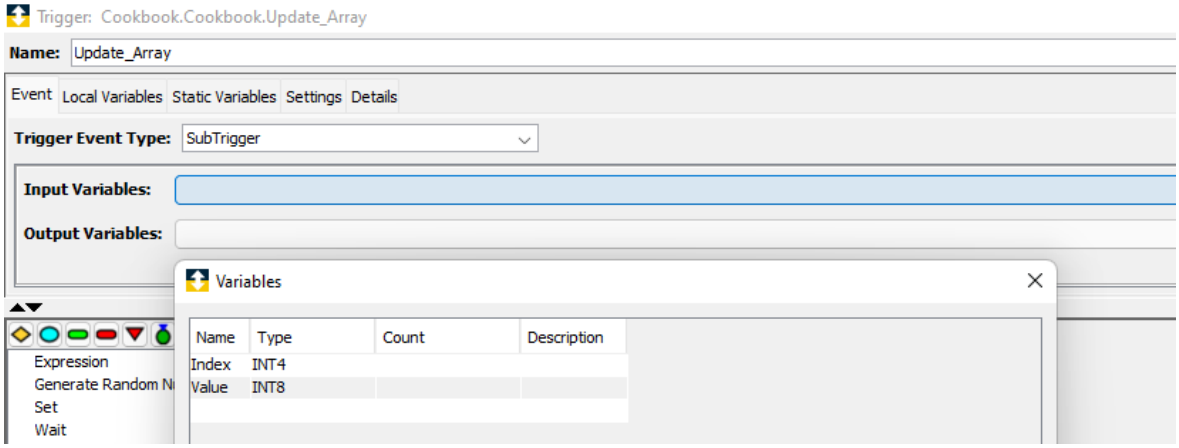
2.4.2. Insert 3 rows containing different values by clicking 'Insert Row'. These rows will be seen and interacted with later in deviceWISE VIEW.

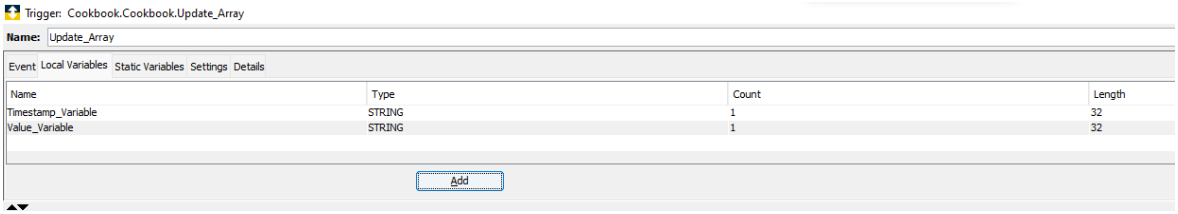


2.5. Creating SubTriggers

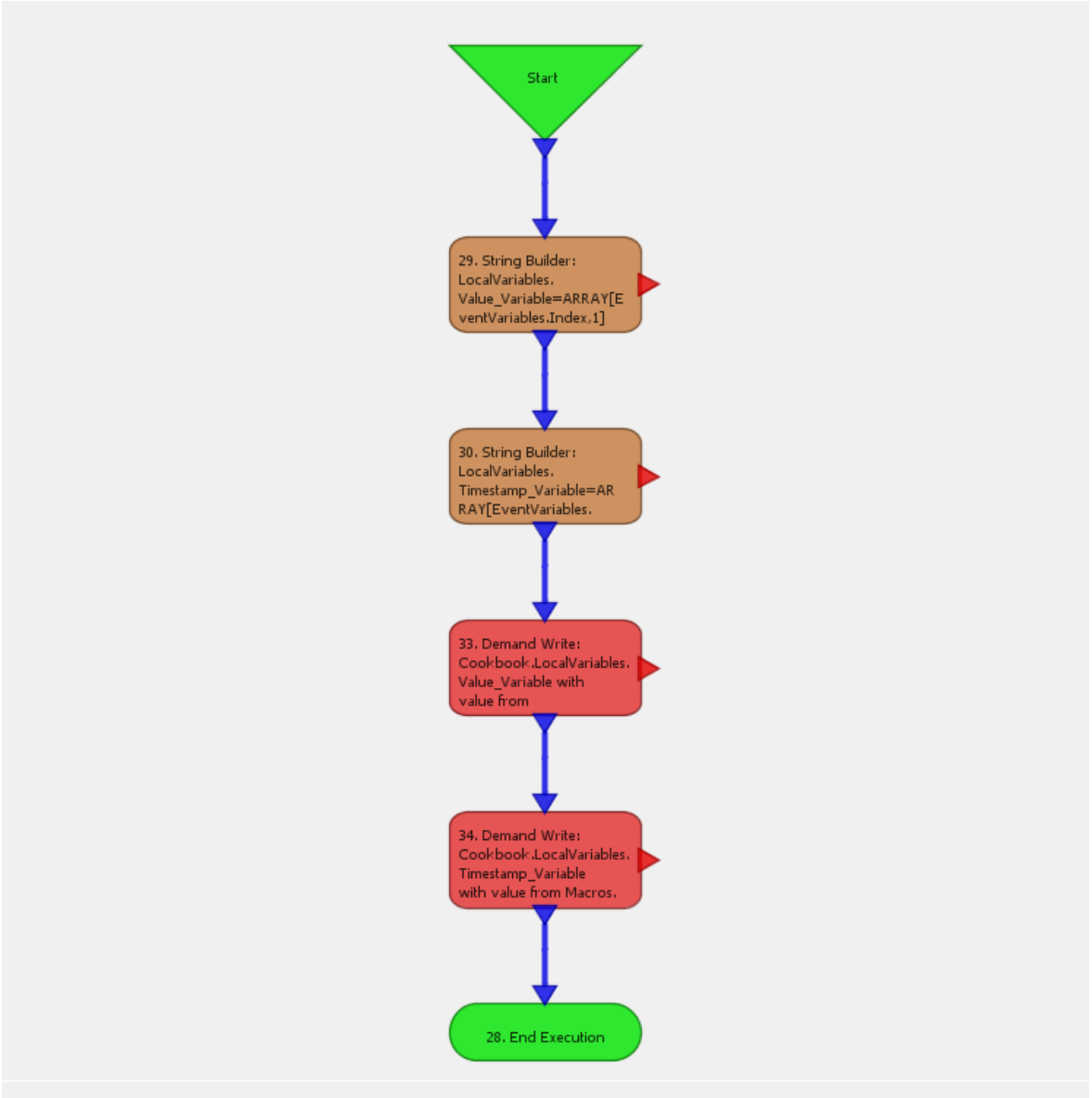
deviceWISE VIEW allows a user to fire triggers and subtriggers using a button widget. To do this, we must first create a new trigger and specify the input variables we want to pass from deviceWISE VIEW.

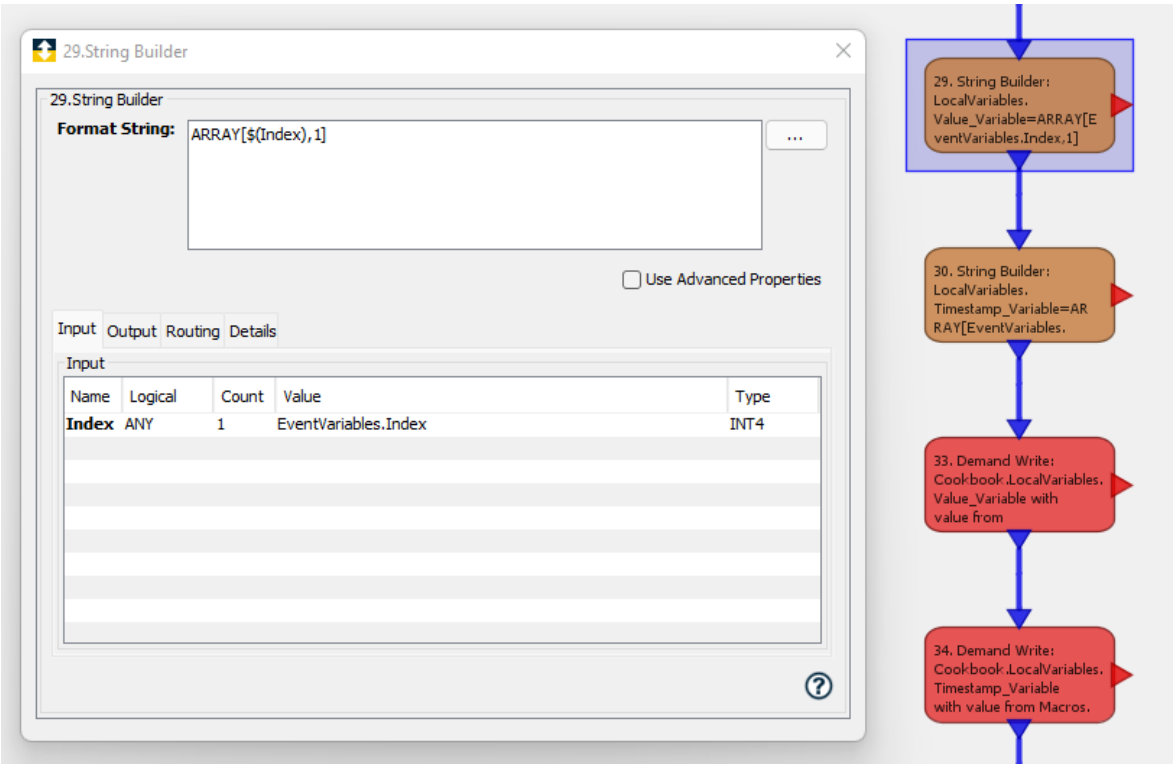
2.5.1. In the Workbench, navigate to 'Projects' from the left panel and create a 'New' project. Rightclick and 'Start' the project. Enter the new project and create a new trigger. Name the trigger "Update_Array" and change the 'Trigger Event Type' to 'SubTrigger'. 'Configure' the input variables and define two (2) INT variables, 'Index' and 'Value'. Navigate to the 'Local Variables' tab under the trigger title and create two STRING local variables, 'Timestamp_Variable' and 'Value_Variable'. Reference the images below.

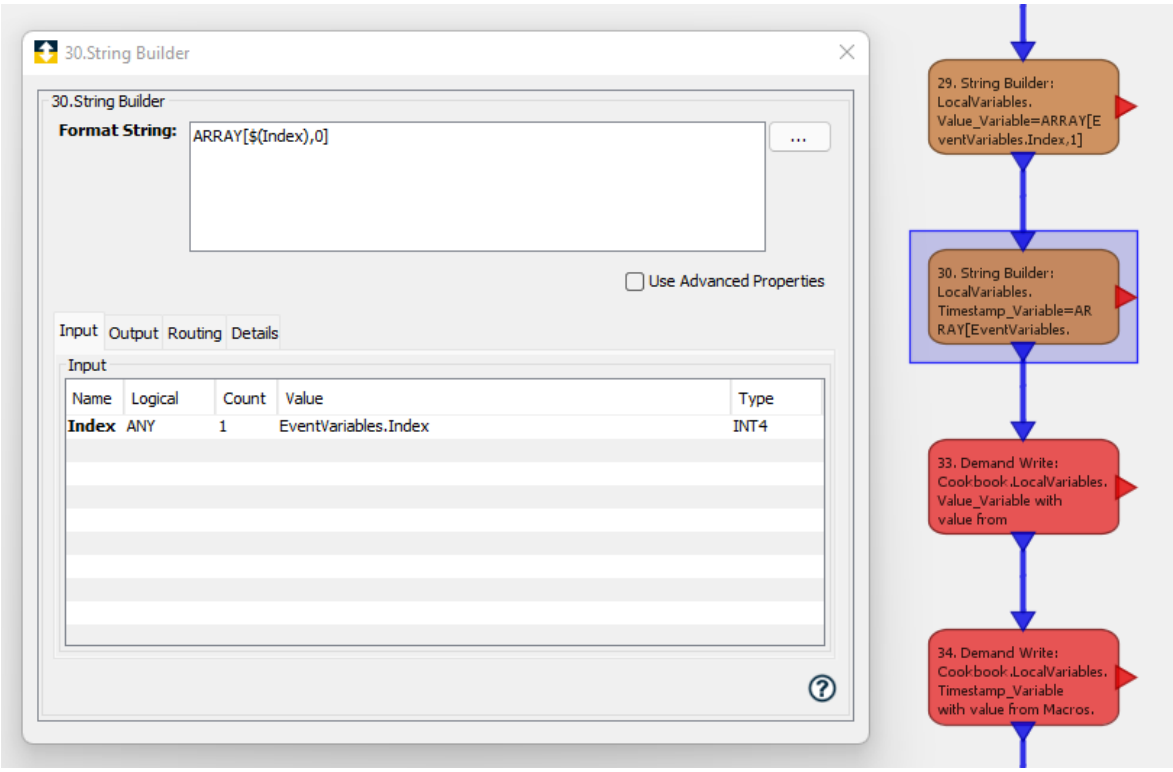


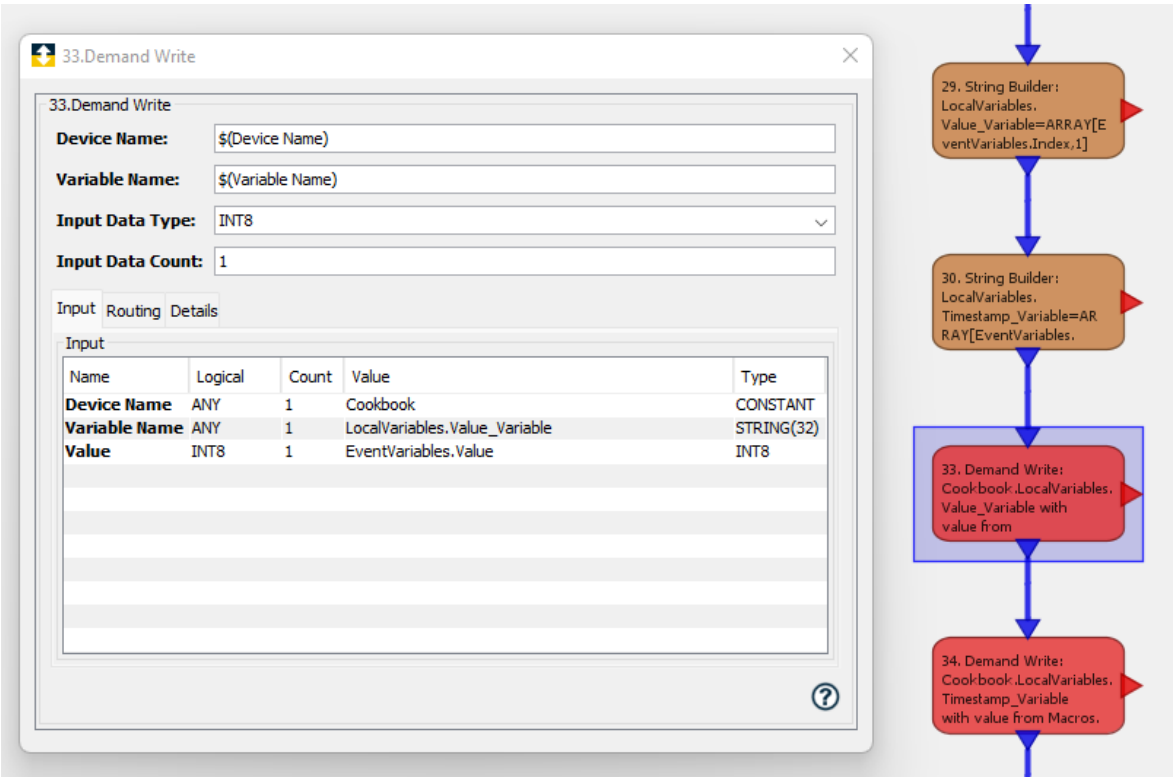


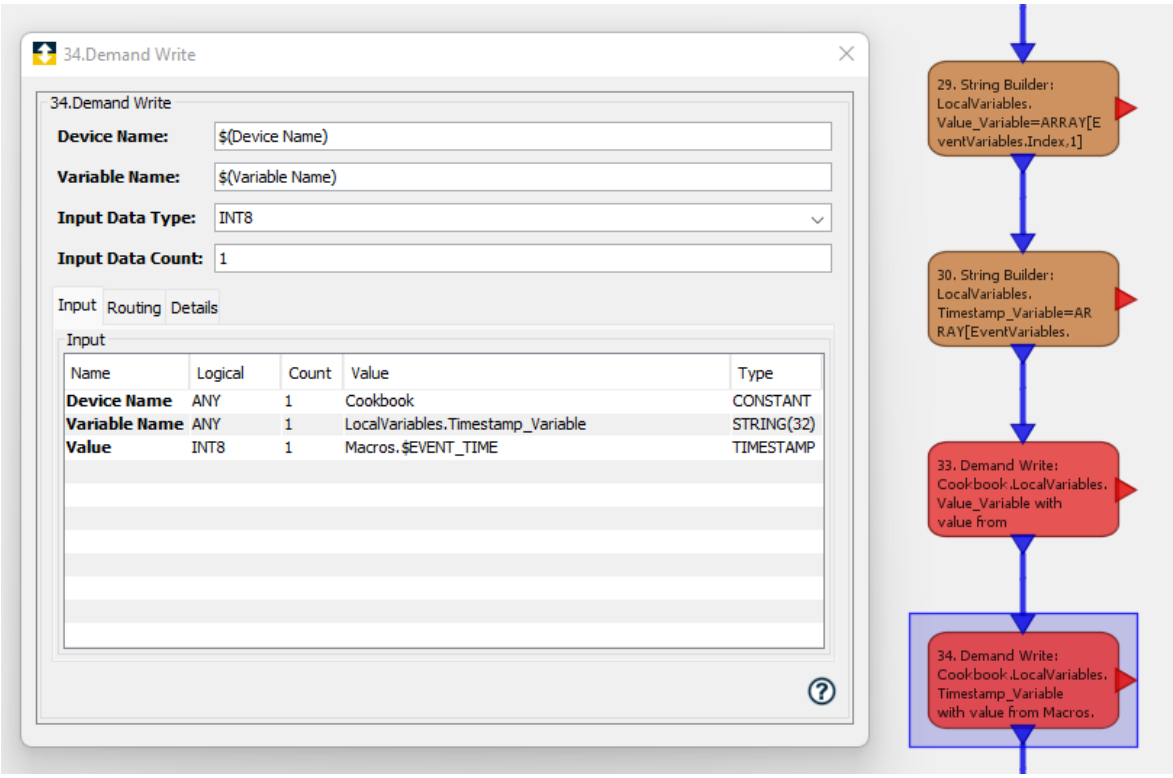
2.5.2. Build the trigger as shown. This will essentially take two (2) input variables passed from deviceWISE VIEW, an integer value and the index in which to store that value in our 'ARRAY' array created in '2.3. Creating Variables'. An associated timestamp will also be inserted along with each value to allow for the display of data over time.



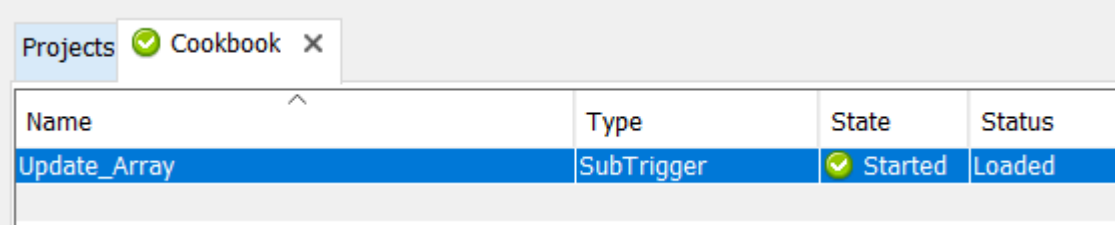

29.String Builder
29.String Builder
Format String:
ARRAY[$(Index),1]
Use Advanced Properties
Input
Output
Routing
Details
Input
Name
Logical
Count
Value
Type
Index
ANY
1
EventVariables.Index
INT4
29. String Builder: LocalVariables. Value_Variable=ARRAY[EventVariables.Index,1]
30. String Builder: LocalVariables. Timestamp_Variable=ARRAY[EventVariables.
33. Demand Write: Cookbook.LocalVariables. Value_Variable with value from
34. Demand Write: Cookbook.LocalVariables. Timestamp_Variable with value from Macros.


30.String Builder
30.String Builder
Format String:
ARRAY[$(Index),0]
Use Advanced Properties
Input
Output
Routing
Details
Input
Name
Logical
Count
Value
Type
Index
ANY
1
EventVariables.Index
INT4
29. String Builder: LocalVariables. Value_Variable=ARRAY[EventVariables.Index,1]
30. String Builder: LocalVariables. Timestamp_Variable=ARRAY[EventVariables.
33. Demand Write: Cookbook.LocalVariables. Value_Variable with value from
34. Demand Write: Cookbook.LocalVariables. Timestamp_Variable with value from Macros.

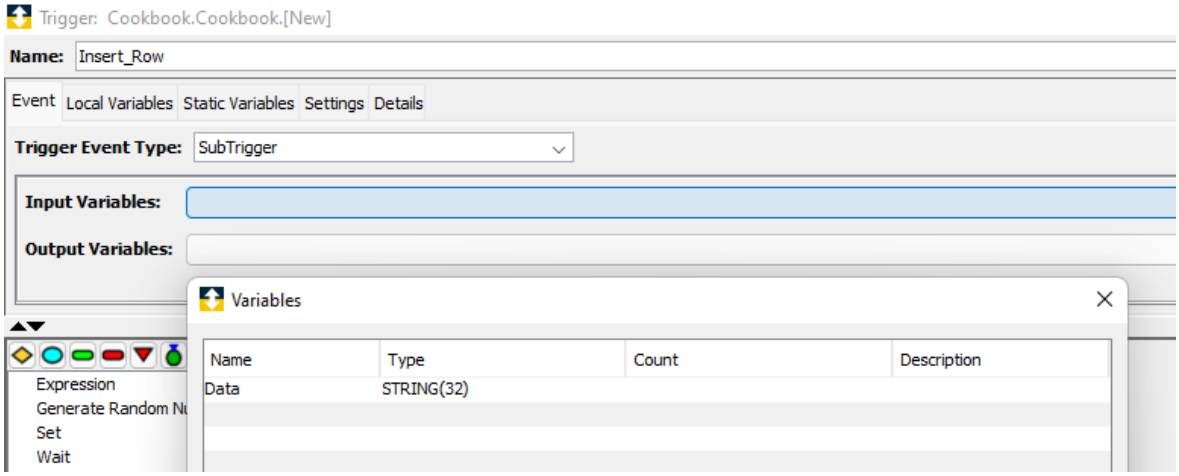




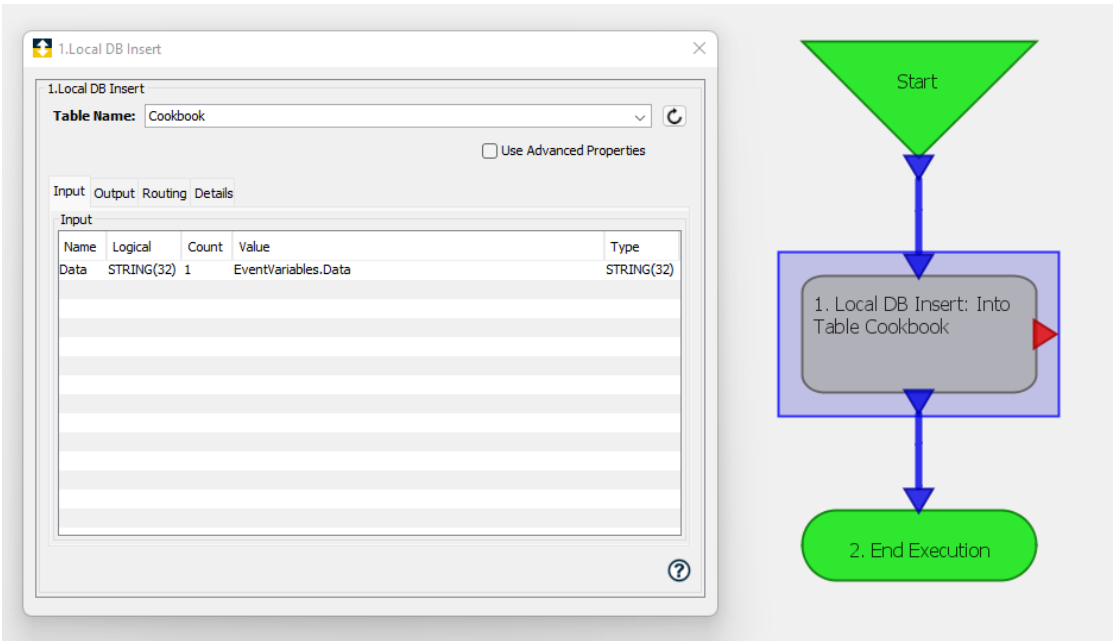
2.5.3. Once completed, ‘Validate’ and ‘Save’ the subtrigger. Start the subtrigger by right clicking on ‘Update_Array’ and clicking ‘Start’.



2.5.4. Within the same project, create a another subtrigger and title it "Insert_Row." Specify one input variable called "Data" of type STRING.



2.5.5. Build the subtrigger as shown. This subtrigger will take an input from deviceWISE VIEW and insert a new row within the 'Cookbook' database table that we created in '2.4. Creating a Local Database Table'. Be sure to select the correct 'Table Name' and use the event variable "Data" as the value being inserted. 'Validate', 'Save', and 'Start' the subtrigger.



3. LOGGING INTO DEVICEWISE VIEW

For further reference content and up-to-date information regarding deviceWISE VIEW, please access our online deviceWISE help site at:

http://docs.devicewise.com

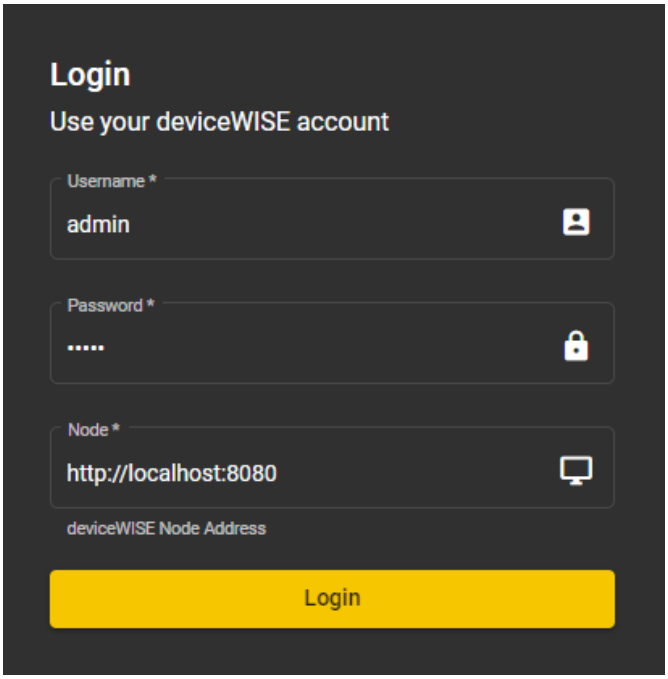
3.1. Accessing the Home Screen

Now that the required deviceWISE VIEW package has been installed on your gateway and the necessary variables and triggers have been created, you may now log in via your web browser and create your first display.

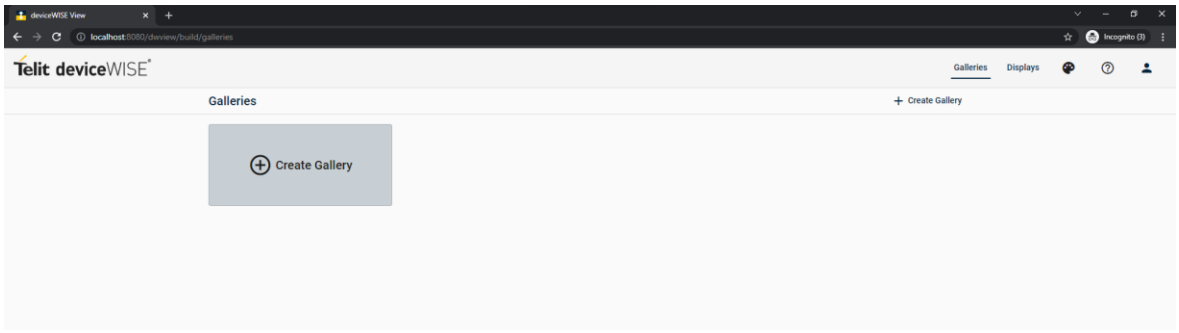
3.1.1. Open your web browser and navigate to your gateway's IP followed by ":8080/dwview". If you are running deviceWISE on your local computer, use the IP 127.0.0.1 or localhost.

If you are remotely connected to your gateway via the Telit IoT Portal, you must first launch a tunnel manager containing HTTP port 8080 to access deviceWISE VIEW. After opening the tunnel, input "127.0.0.1:8080/dwview" as the URL in your web browser.

3.1.2. When met with a login screen, enter the default 'admin' for both credentials and log in.



3.1.3. You should now see the deviceWISE VIEW home screen where you can create, manage, and edit your galleries and displays.



4. CREATING YOUR FIRST DISPLAY

Now that you are logged into deviceWISE VIEW, we can start building displays and utilizing widgets to visualize data.

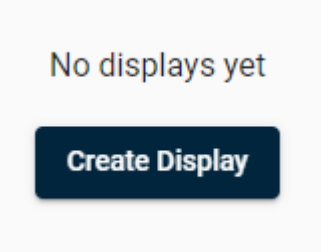
For further reference content and up-to-date information regarding deviceWISE VIEW, please access our online deviceWISE help site at:

http://docs.devicewise.com

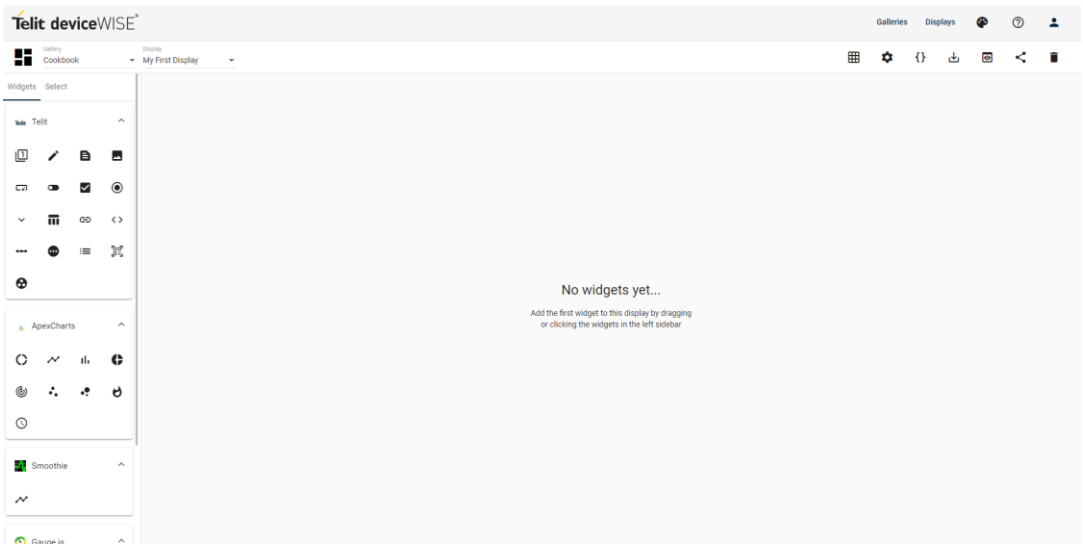
4.1. Entering a Display

To create a new display, we must first make a new gallery. Galleries are essentially folders that house a collection of displays.

4.1.1. Click on ‘Create Gallery’ and name it. For the sake of this guide, we will be naming it ‘Cookbook’. ‘Create’ and enter the new gallery. Create a new display by clicking ‘Create Display’. Name the new display “My First Display”.



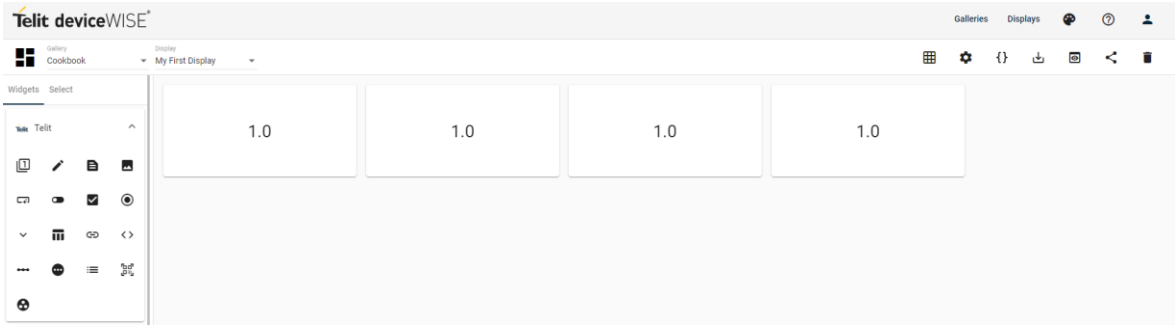
4.1.2. Entering the display brings up the deviceWISE VIEW workspace where you can interact with various widgets and tools.



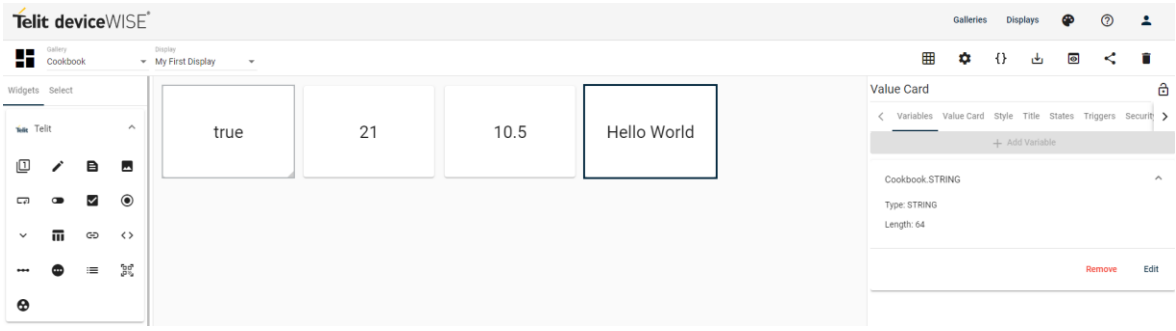
4.2. Reading Data in deviceWISE VIEW

Since we have already created variables on the Workbench that deviceWISE VIEW can access, we can start visualizing them using widgets.

4.2.1. Click or drag four (4) 'Value Card' widgets onto the display. These value cards will be reading from the variables created in section '2.3. Creating Variables'.



4.2.2. To configure the value card widgets, click on them so the configuration panel appears from the right of the screen. Here you can assign the variable that each value card will read from by locating the 'Variables' tab and clicking 'Add Variable'. Specify the 'Cookbook' global variable device and select a variable. Do this for each of the variables we created, excluding the array. The data should immediately populate the value cards.

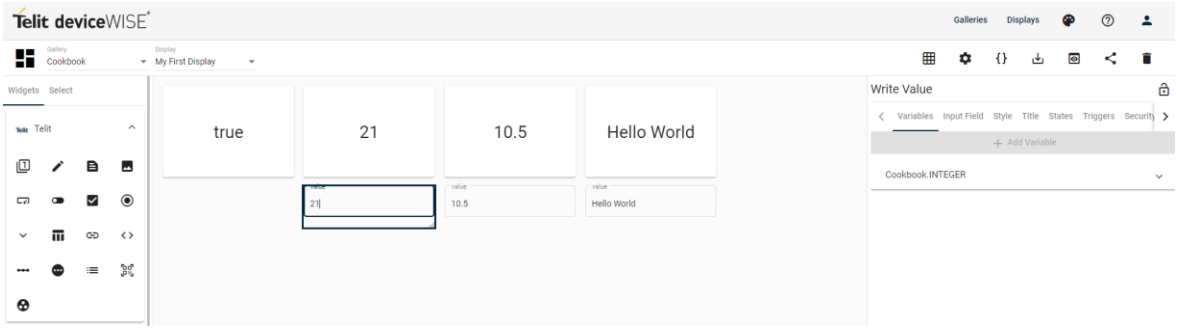


It is that simple to read live data from any device using deviceWISE VIEW. Any device that enumerates variables such as a PLC, Modbus device, the System Monitor, or a global variable group can be seen and interacted with. This also means any trigger logic that writes to a global variable can be read. deviceWISE VIEW cannot access local or static trigger variables.

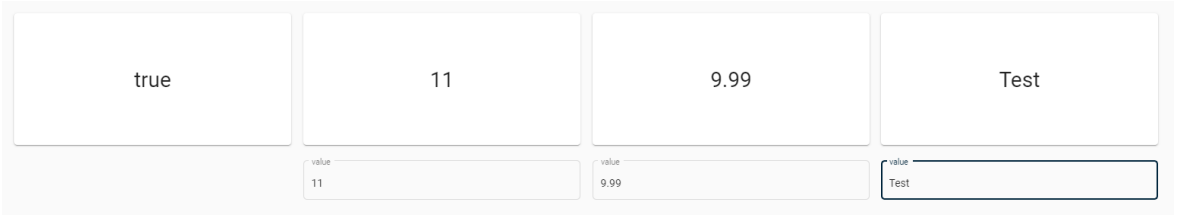
4.3. Writing Data in deviceWISE VIEW

While it is easy to read data using deviceWISE VIEW, it is also important to be able to write to variables, enabling the ability to toggle outputs on a controller or write to a specific Modbus register.

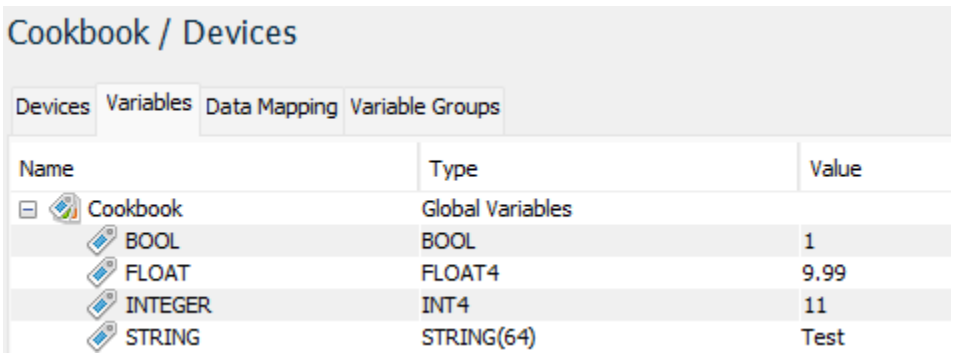
4.3.1. Click or drag three (3) 'Write Value' widgets on to the display. Assign the widgets to the same variables, excluding the BOOL variable, and organize them neatly under the value cards they are "writing" to.



4.3.2. Save any changes and click on the eye icon near the top right of the screen labeled 'View Live' to enter the live display. Write to the global variables using the write value widgets by typing in the 'value' field. The value cards above them should update instantly as they read from the same variables being written to.



4.3.3. Visit the 'Variables' tab within the 'Devices' section of the Workbench and read the variables after writing to them.

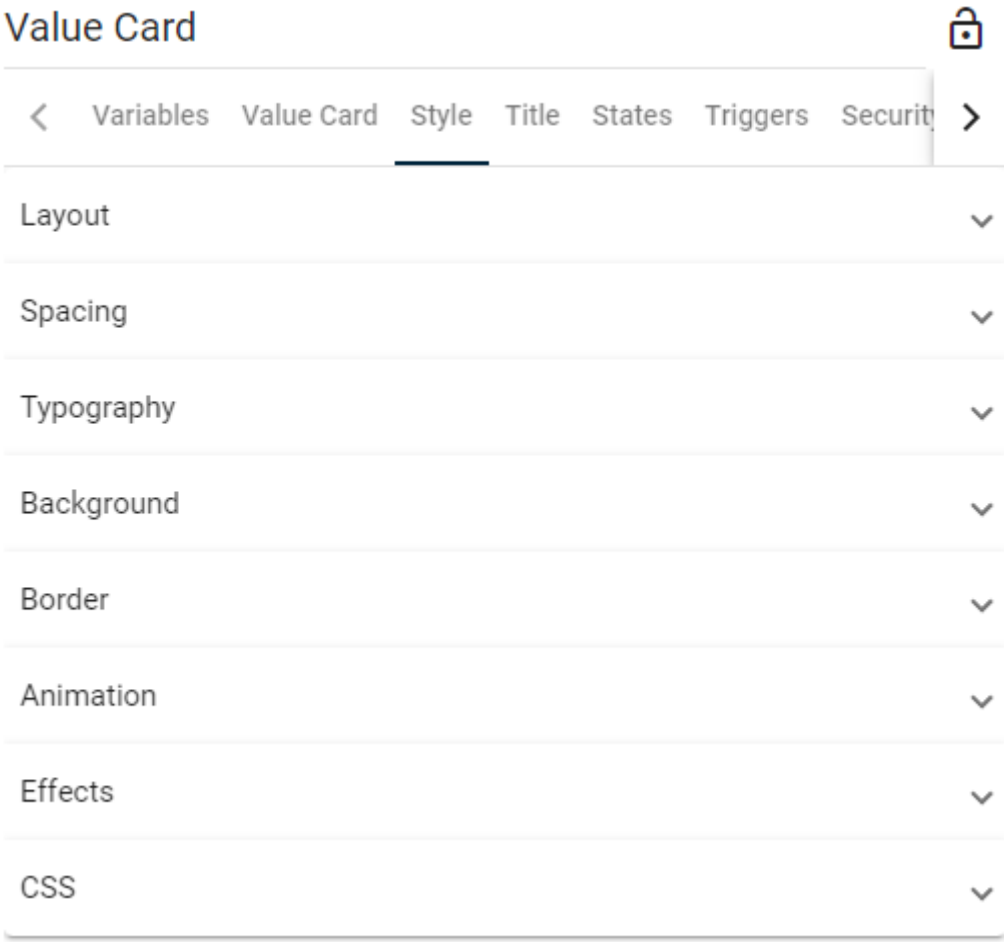


4.4. Styling Widgets

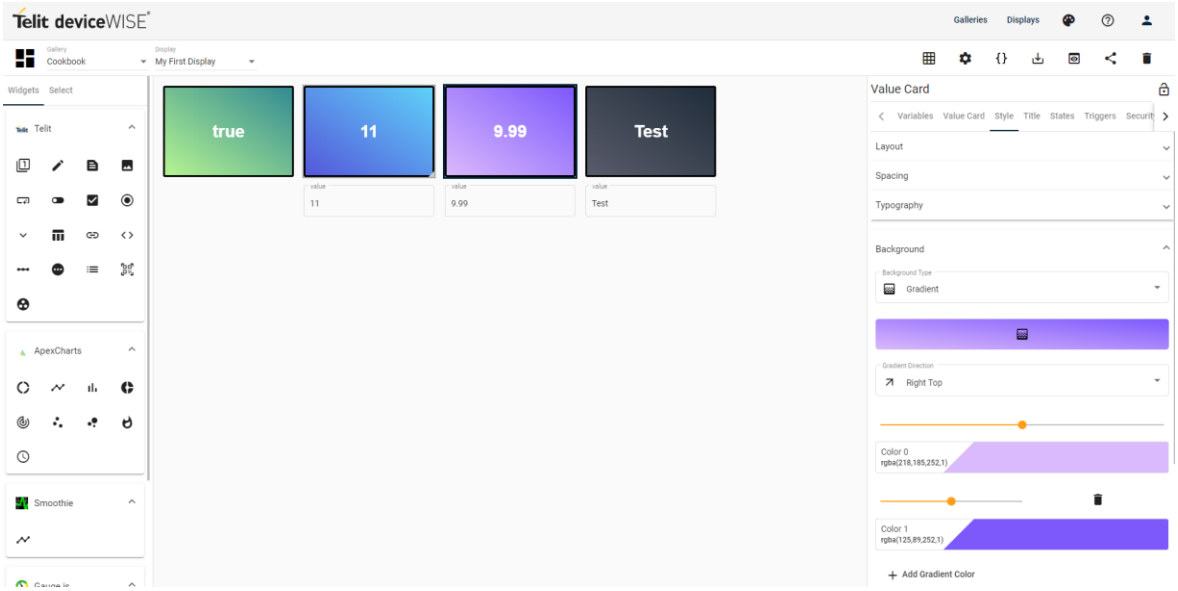
You will probably notice that it is a little hard to see the widget boxes over a light background. Luckily enough, deviceWISE VIEW has a suite of styling features that allow a user to completely personalize their displays and widgets. These styling features are available on every widget.

4.4.1. Enter edit mode again by clicking the 'X' at the top right of your live display and selecting 'Edit My First Display'.

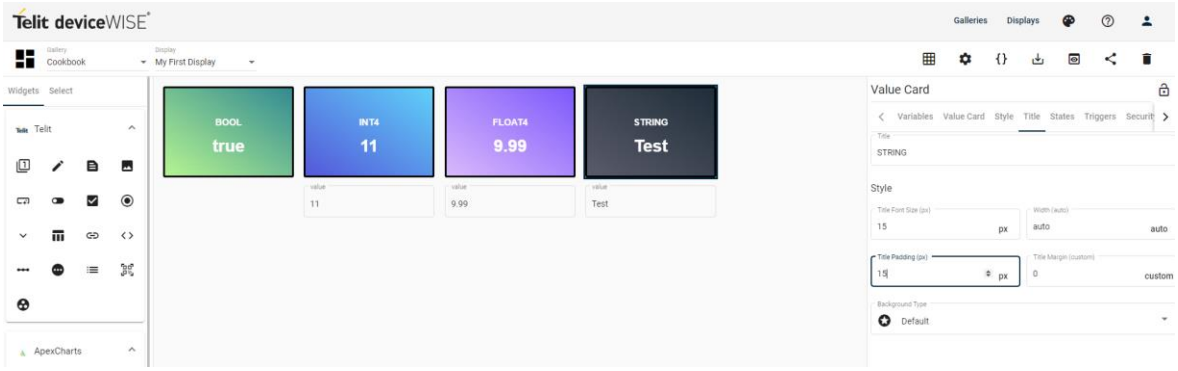
4.4.2. Click on any widget you would like to edit to open the widget configuration panel. Navigate to the 'Style' tab. Here you can style the widget by adding backgrounds, borders, animations, and more.



The example below shows the widgets configured with a bolded Helvetica font from under 'Typography', a white font color, several preset gradient backgrounds from under 'Background', and a 3 px solid black border from under 'Border'.



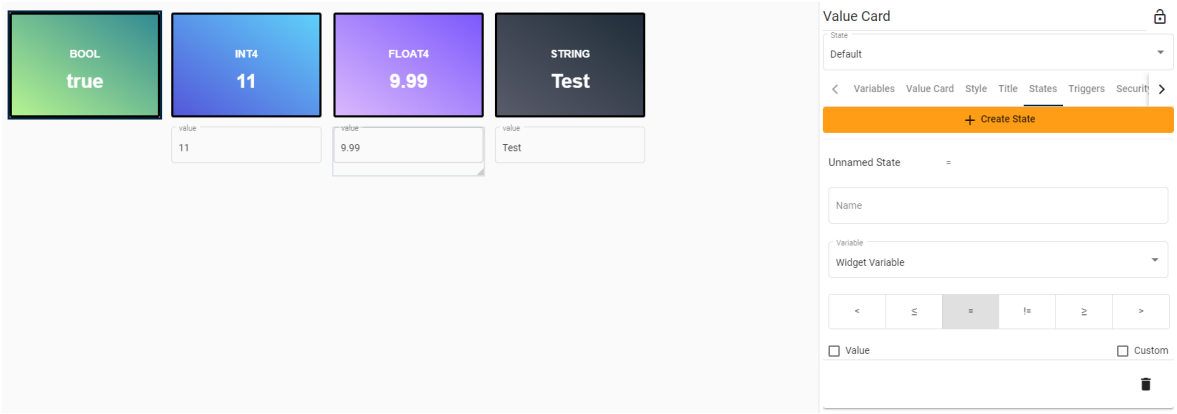
4.4.3. You can also add a title to the widgets to make displays easier to read and navigate. Click on the 'Title' tab within the widget configuration panel and give each widget a title based on the variable they are reading. Adjust the settings to your preference.



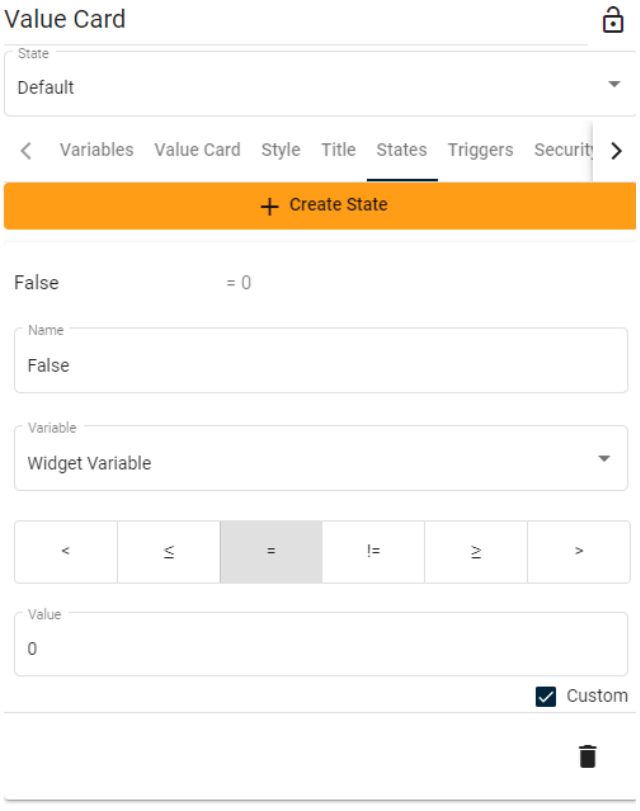
4.5. Using States

Another useful feature within deviceWISE VIEW is the ability to define 'States', allowing a user to alter a widget's styling, security, and trigger settings when a user-defined variable satisfies an expression. For this guide, we will be defining a state that changes the BOOL value card color when the BOOL variable becomes true or false. This is useful for displaying alarms when variable values change. It can also be used for animating, which will be covered later in the guide.

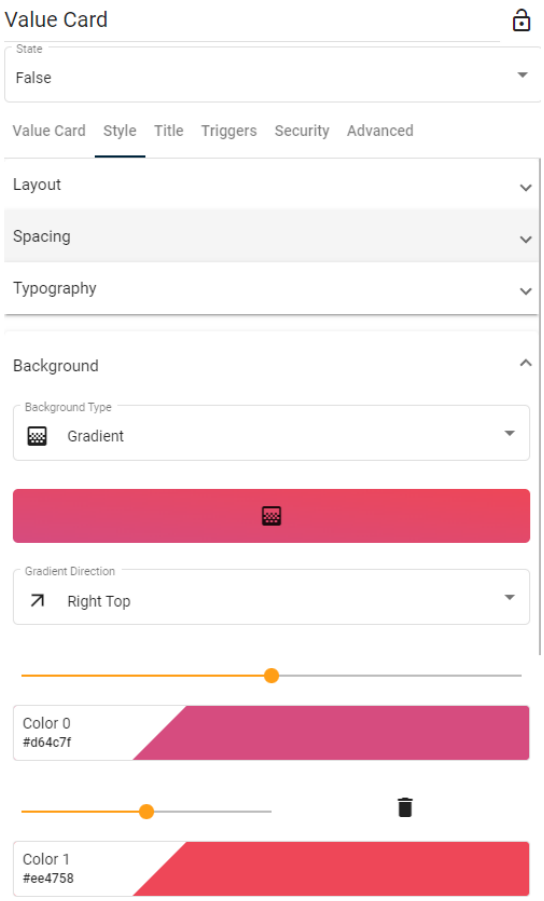
4.5.1. Select your BOOL value card widget to enter the widget configuration panel. Navigate to the 'States' tab and create a new state by clicking 'Create State'.



4.5.2. Name the state “False” and check the ‘Custom’ checkbox. Enter “0” for ‘Condition Value’. This will cause the widget to change state whenever the widget variable, which in this case is the BOOL variable that it is reading, is equal to zero (0), or ‘false’. Reference the image below.

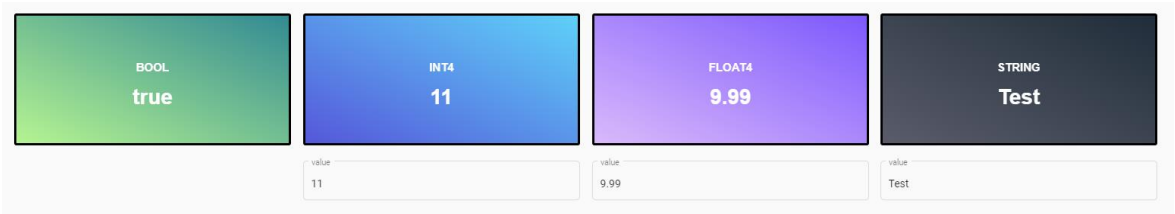


4.5.3. Navigate back to the ‘Style’ tab within the same widget and select our newly created state ‘False’ from the ‘State’ drop-down menu at the top. Any styling edits made when a state is selected will only show when that state is triggered, or in this case when the widget value becomes zero (0). For this guide, we will be changing the background color of the widget to a red gradient whenever the state is triggered.

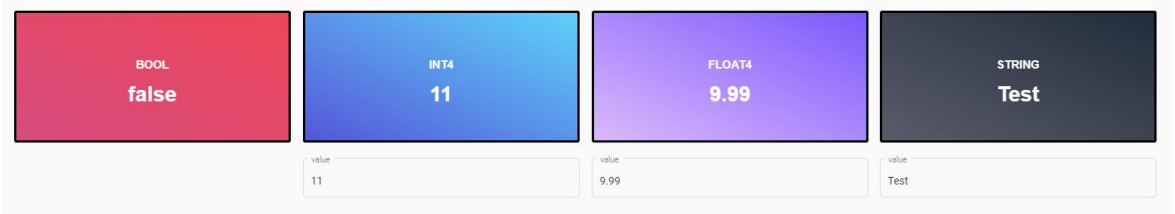


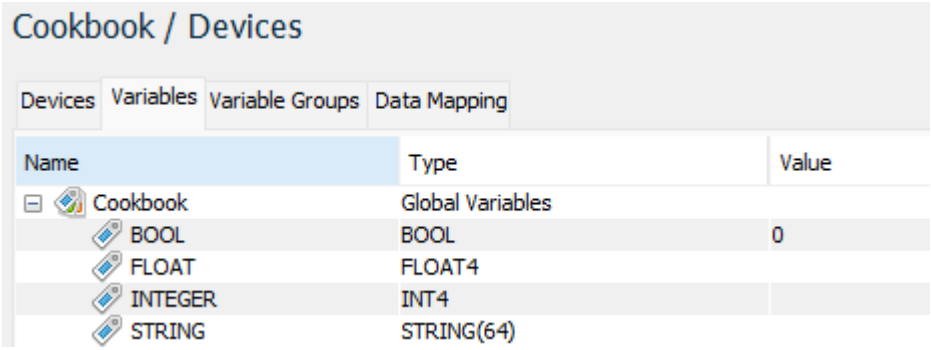
4.5.4. Save any changes and click the eye near the top right of deviceWISE VIEW to enter the live display. If we now write the value zero (0) to the BOOL global variable on deviceWISE, the BOOL value card widget will turn red as the state is triggered. When the value is not equal to zero (0), the widget will return to its 'Default' styling.

Default State



Triggered State

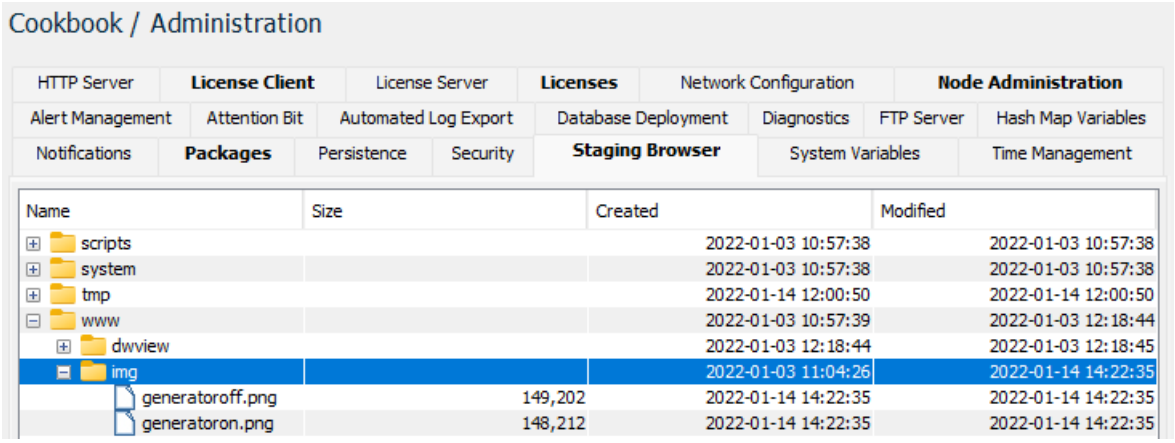




4.6. Uploading Custom Images

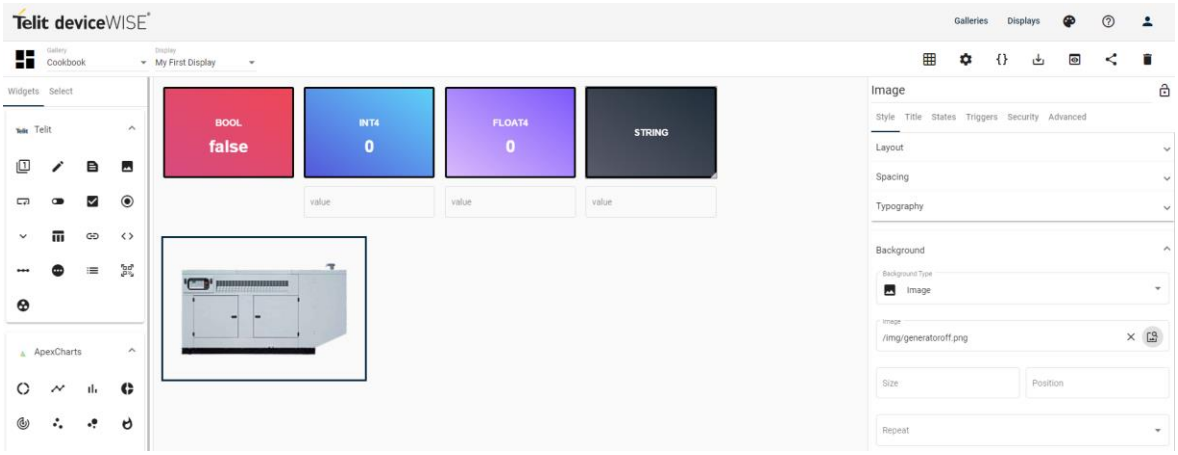
To completely personalize your display, deviceWISE VIEW allows a user to upload images directly to the deviceWISE Staging Browser for immediate access.

4.6.1. On the Workbench, navigate to 'Administration' from the left panel of your node. Click on the 'Staging Browser' tab and navigate to the 'img' folder within the 'www' directory. This is where images can be uploaded and accessed via deviceWISE VIEW. To upload an image, simply right click on the 'img' folder and select 'Put'. Upload at least two (2) custom images to work with. For this guide, I will be working with a generator image and animating it on and off.

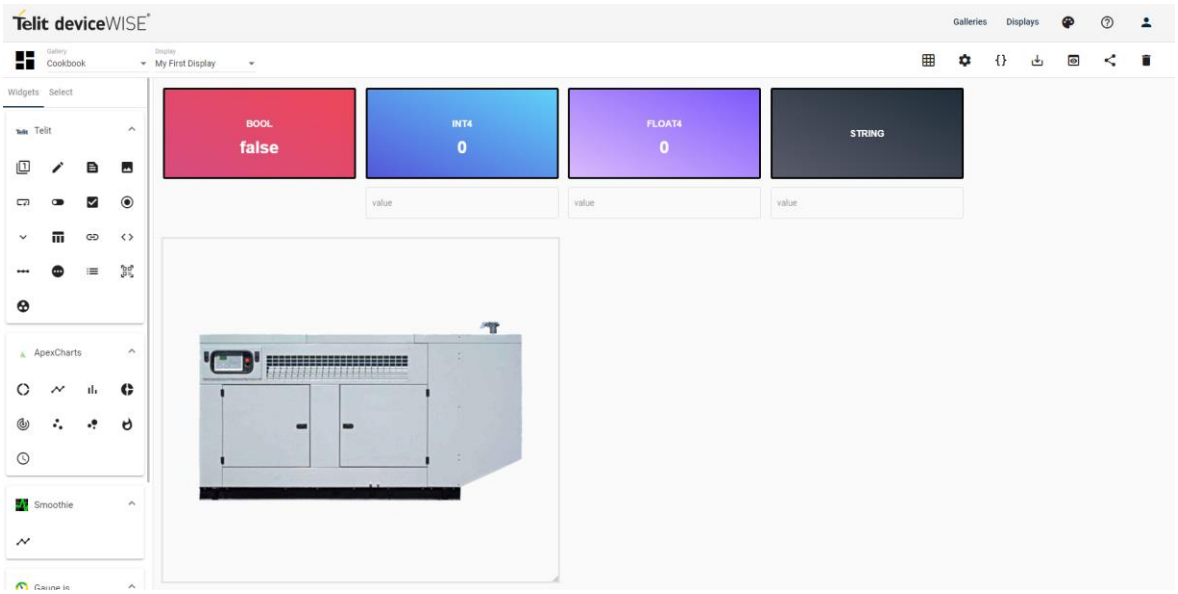


4.6.2. In deviceWISE VIEW, drag the 'Image' widget onto your display. Image widgets are dedicated to encapsulating images and do not require an assigned variable.

4.6.3. Select one of the images uploaded to the Staging Browser by opening ‘Background’ within the widget configuration panel and choosing ‘Image’ as the ‘Background Type’. To select your image, click on the find image icon next to ‘Image’ and select the image you want to be displayed within the image widget.



4.6.4. Set the ‘Size’ to ‘Contain’ to contain the image within the widget boundaries. You can resize the image by clicking and dragging one of the corners.



4.6.5. If you would like, continue modifying the image by selecting its position within the widget, the background color, or any of the other styling options available.

5. USEFUL DEVICEWISE VIEW TOOLS

For further reference content and up-to-date information regarding deviceWISE VIEW, please access our online deviceWISE help site at:

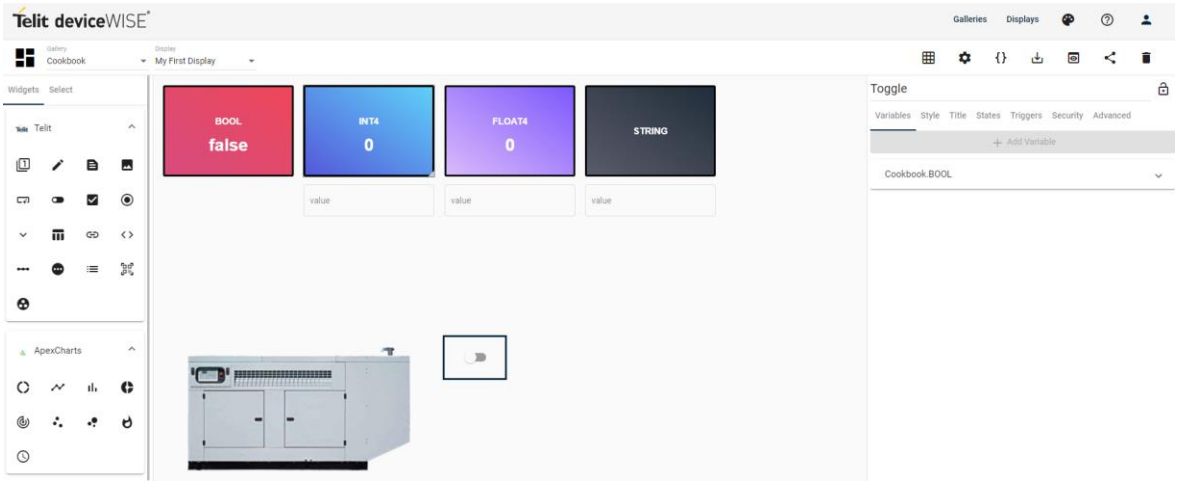
http://docs.devicewise.com

5.1. Using Animation

Using states, different images can be displayed within the same image widget due to the fact that everything is changed within the styling section. This can allow for animation. Recall in '4.5 Using States' that states allow a user to change components of a widget's configuration when a user-defined variable satisfies an expression. For this we will be using a 'Toggle' widget to toggle the BOOL global variable we defined earlier in '2.3. Creating Variables'. This will serve as an on/off switch for the "generator".

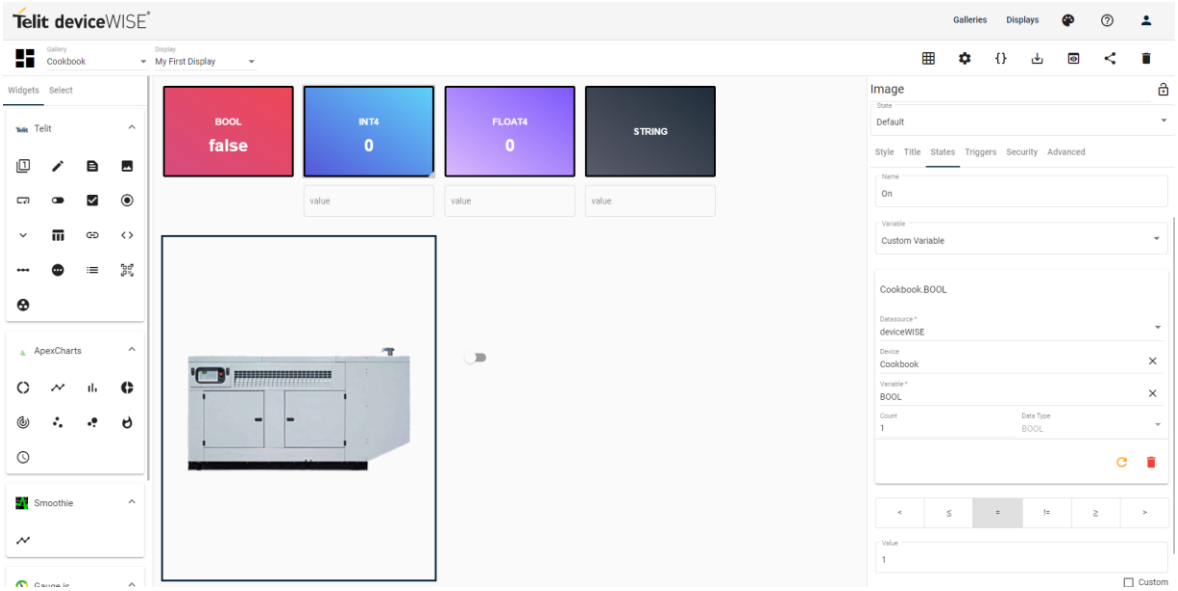
5.1.1.

Drag a 'Toggle' widget onto the display next to your image and click on it to open the widget configuration panel. Specify the BOOL variable from the 'Cookbook' device to toggle on or off, or '1' and '0'.

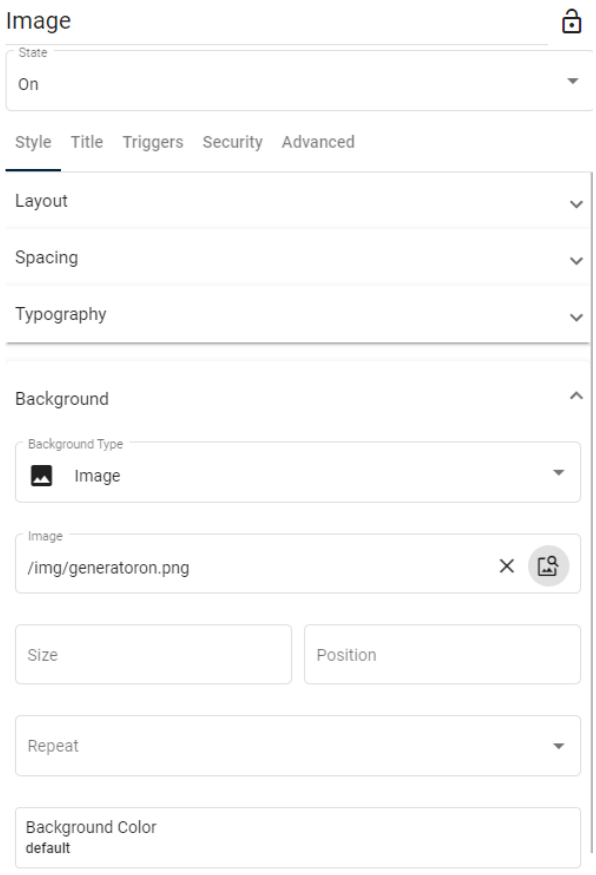


5.1.2.

Enter the configuration for the image widget and navigate to the 'States' tab. Here we will define a new state called "On" that reads the same BOOL variable and will be enabled once it is equal to 1.



5.1.3. To animate the image, navigate to the ‘Style’ tab from within the image widget configuration. From the ‘State’ drop-down above it, select our newly created ‘On’ state. Select a new image within the ‘Background’ section using the same method previously described in ‘4.6. Uploading Custom Images’. You will notice that the displayed image does not immediately change, and that is because the state has not been triggered yet.



5.1.4. Save and enter the live display. When the toggle widget is switched, the image should now change from the original image you selected to the new image set within the 'On' state styling options. When I toggle the BOOL variable, the generator turns 'on' or 'off'. You will also notice that the BOOL value card widget also changes state because it is reading from the same BOOL variable being toggled.

Default State

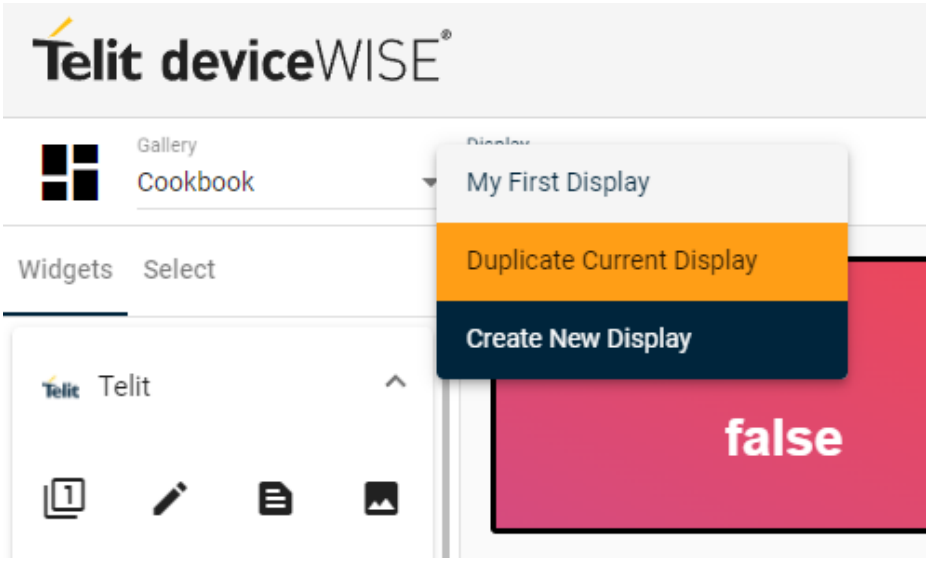
Triggered State

While there is no generator actually turning on or off in this example, this could essentially be deployed by reading real device variables such as their status and creating states based off of that information. Toggle widgets can also write to start bits to remote start devices.

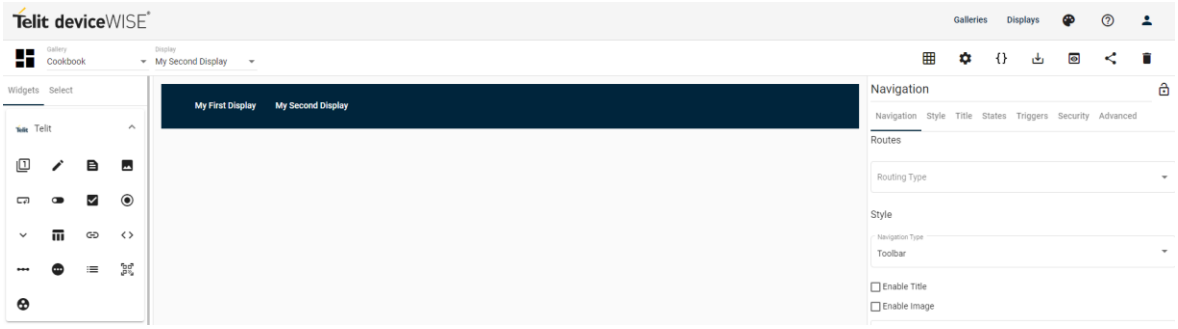
5.2. Using Navigation

In a situation that requires multiple displays, each display can access another via navigation. There are two (2) ways of implementing navigation: the 'Navigation' widget, or 'Button' widgets.

5.2.1. Create a new display by clicking the display drop-down near the top left of the window and selecting 'Create New Display'. Name this new display "My Second Display."

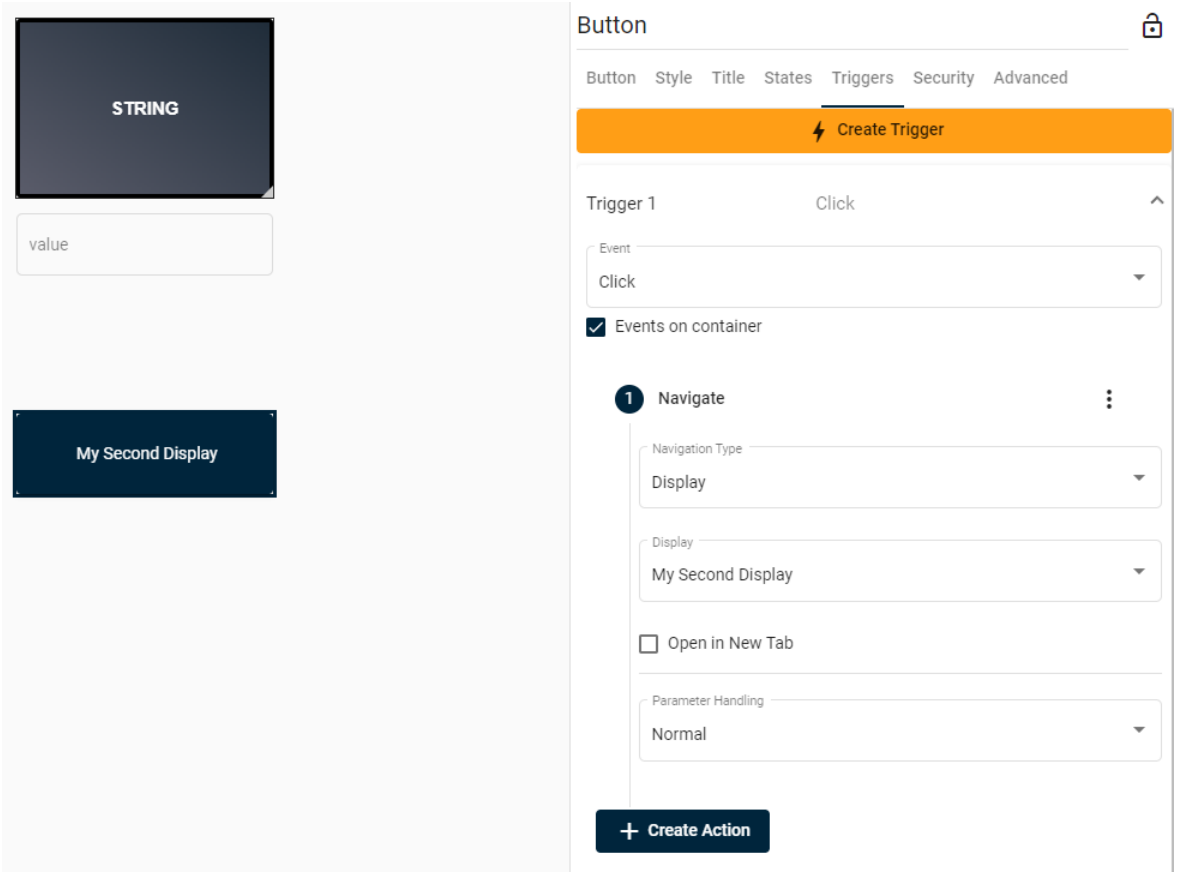


5.2.2. In the new display, drag a 'Navigation' widget to the top of the screen. The widget will discover any available displays within the gallery and automatically populate the widget.



5.2.3. Go back to ‘My First Display’ and drag a ‘Button’ widget onto the workspace. Click on it to enter the widget configuration and set the ‘Button Text’ to “My Second Display.”

5.2.4. Navigate to the ‘Triggers’ tab and create a new trigger by clicking ‘Create Trigger’. Open the trigger, click ‘Create Action’, and from the drop-down select the ‘Navigate’ action. Select ‘My Second Display’ as the display destination.

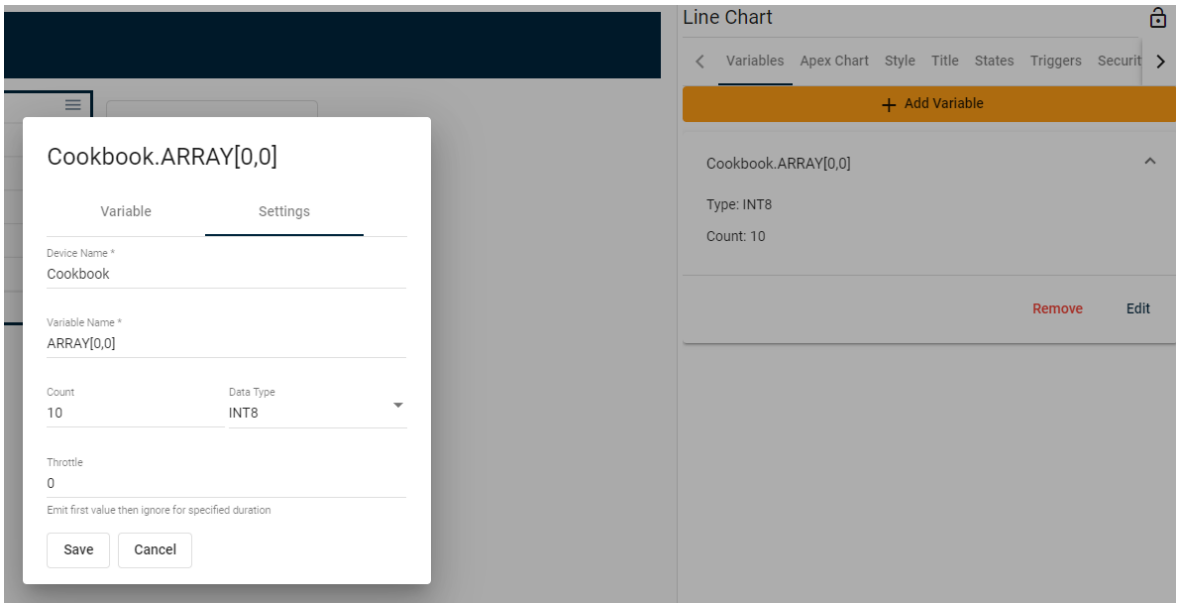


5.2.5. Save and enter the live display. Navigate between the two displays using the newly-created navigation widgets.

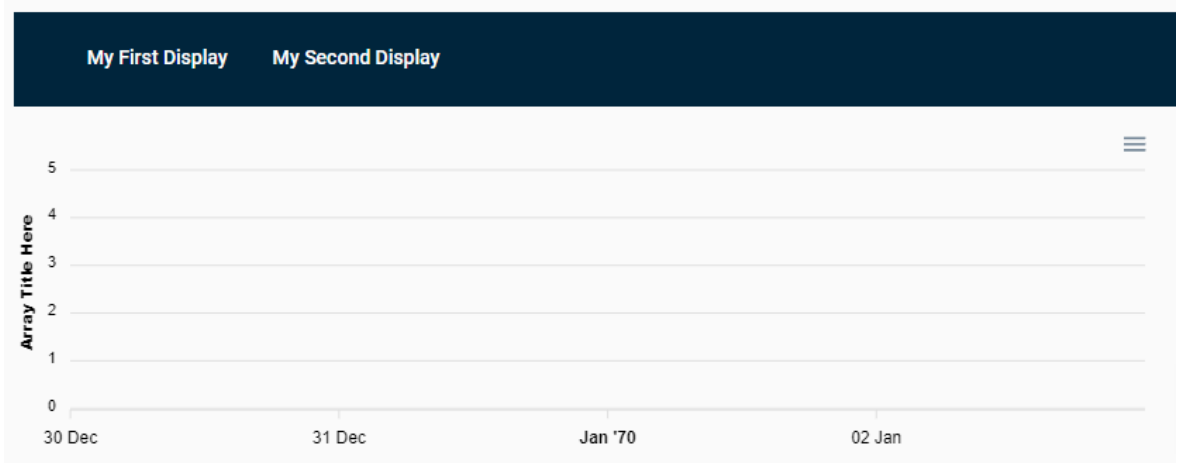
5.3. Implementing a Trend Chart

deviceWISE VIEW uses ApexCharts for configuring and displaying line charts. Trend charts can be implemented using an array defined within the Workbench to display data over time.

5.3.1. In ‘My Second Display’, drag and drop a ‘Line Chart’ widget from under ‘ApexCharts’ onto the display. In the configuration panel and select the global variable ‘ARRAY’ created in ‘2.3. Creating Variables’ at the [0,0] index. Enter the variable settings by clicking ‘Edit’ and ensure that the array ‘Count’ is equal to ten (10) and the ‘Data Type’ is ‘INT8’. The ‘Count’ will always equal the xdim multiplied by the ydim of the array. In our ‘ARRAY’ array, the ‘Count’ is demonstrated as 5 x 2 = 10.



5.3.2. Under the ‘Apex Chart’ tab, navigate to the ‘X Axis’ section and ‘Enable X Axis Settings’. Change the ‘Type’ to ‘Date Time’ to the timestamp data of the array is displayed on the x axis. Navigate to the ‘Y Axis’ section and ‘Enable Y Axis Settings’. Scroll down and click ‘Add Axis Title Settings’ and enter a title for the array in the ‘Text’ field.

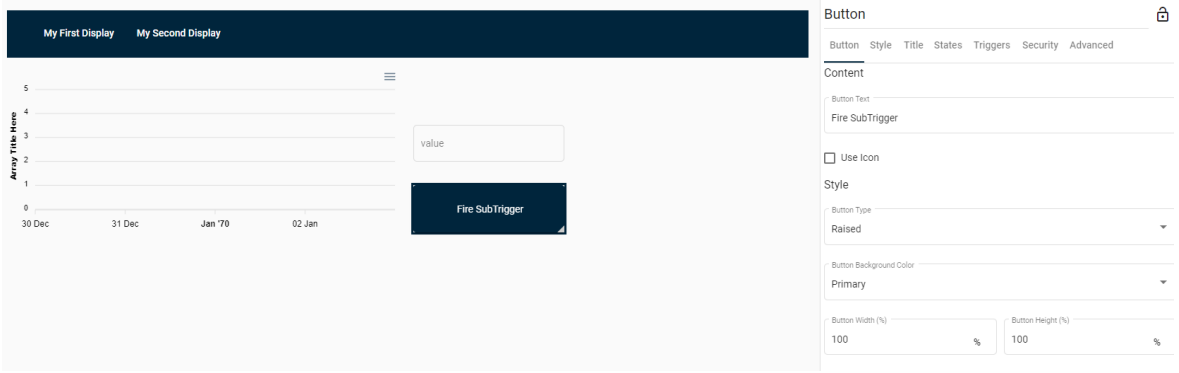


5.3.3. You will notice no data is present within the line chart widget. The following sections will demonstrate how to call our previously created ‘Update_Array’ subtrigger while passing inputs from deviceWISE VIEW into the line chart.

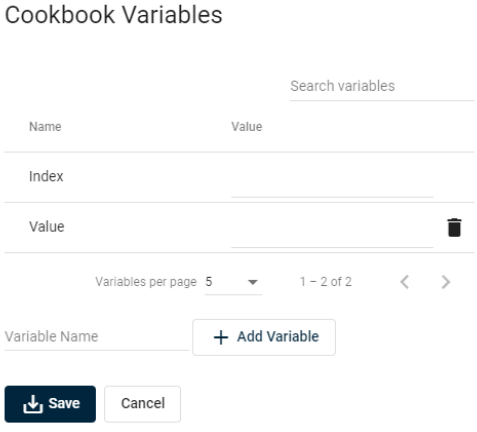
5.4. Calling SubTriggers with Parameters

deviceWISE VIEW allows users to fire triggers using a button widget, but local variables must be utilized in order to forward parameters or inputs directly from a display into a defined subtrigger. deviceWISE VIEW local variables are available across every display within the same gallery.

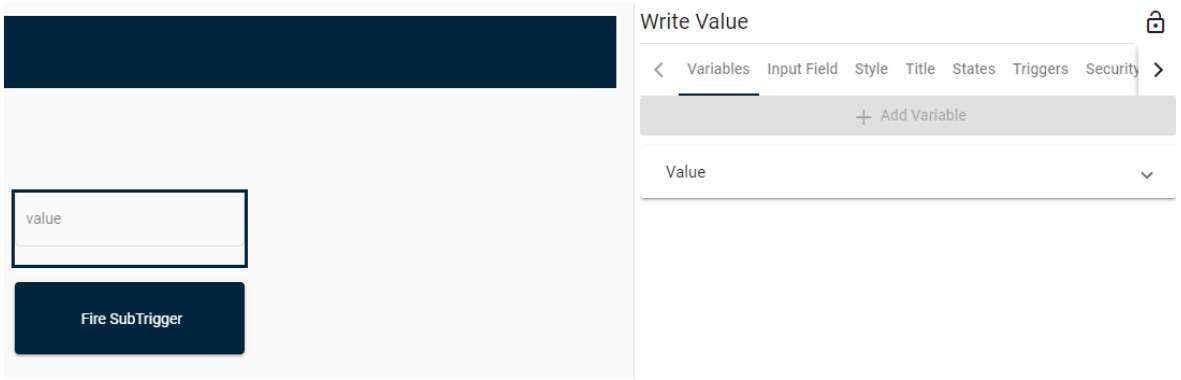
5.4.1. In deviceWISE VIEW, drag a ‘Write Value’ widget onto the display. Directly below it, place a ‘Button’ widget and set the button text to “Fire SubTrigger.” Resize or move the widgets to consolidate space.



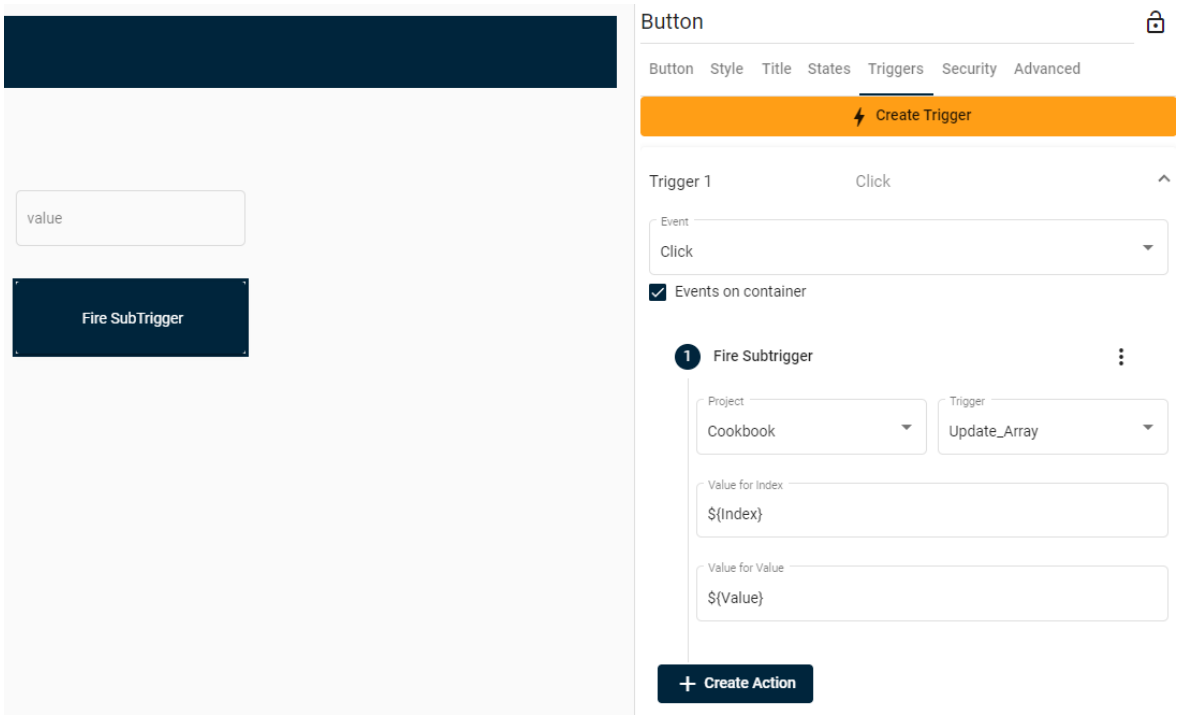
5.4.2. To access the deviceWISE VIEW local variables, click on the curly brackets near the top right of the window labeled ‘Local Variables’. Create two (2) local variables by naming them and clicking ‘Add Variable’. Ensure these match the input variables that were configured for the subtrigger in ‘2.5. Creating SubTriggers’. Reference the image below.



5.4.3. ‘Save’ the local variables. Set the ‘Write Value’ widget to write to the ‘Value’ local variable. You can specify the local variables in the ‘Variables’ tab by changing the ‘Datasource’ to ‘Local’.



5.4.4. To pass the local variable values when firing the ‘Update_Array’ subtrigger created in ‘2.5. Creating SubTriggers’, enter the widget configuration for the button below the ‘Write Value’ widgets. Navigate to the ‘Triggers’ tab and create a new trigger that performs a ‘Fire Subtrigger’ action when clicked. Select the ‘Update_Array’ subtrigger as the trigger to be fired. For each of the parameters, a deviceWISE VIEW local variable must be specified by entering “${variablename}” for the value. Using that format, specify the two local variables created in ‘5.4.2’. This will take a user’s inputs from deviceWISE VIEW and pass them through the input variables defined when creating the ‘Update_Array’ subtrigger. Reference the image below.

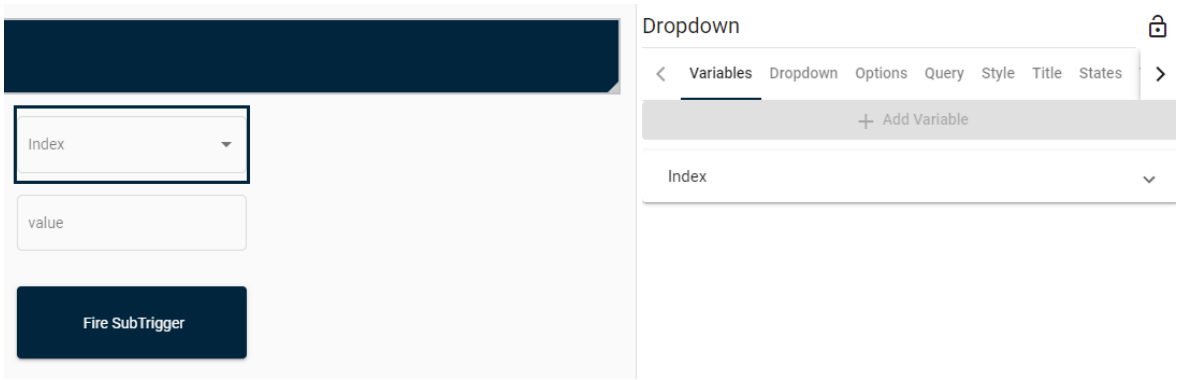


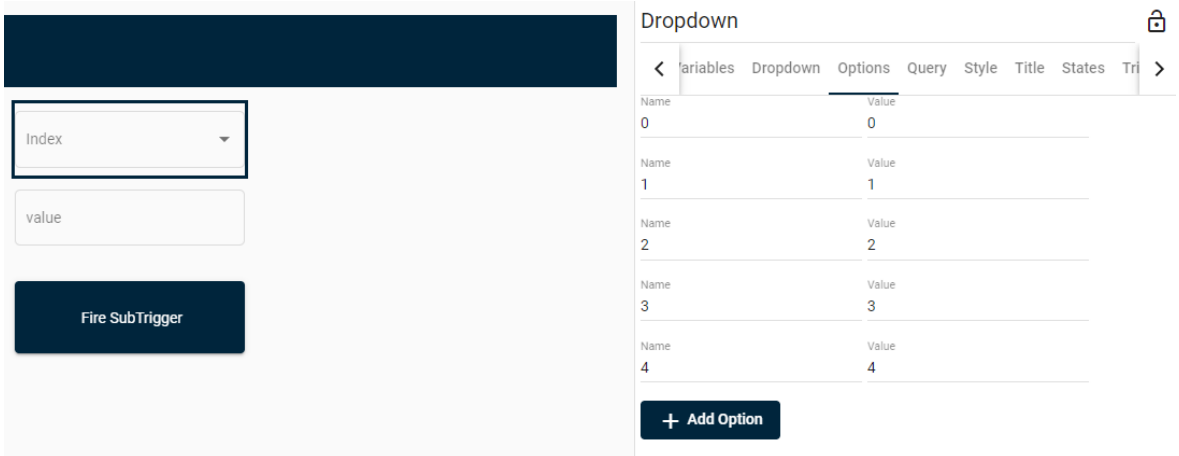
Value parameters in widget configurations are completely dynamic using the ${variablename} format. This allows a user to pull values from both deviceWISE device variables and deviceWISE VIEW local variables. Please note if using deviceWISE device variables, the full name of the variable including the device name and any indexes must be included within the ${variablename} format.

5.4.5. While this handles a user's input for value, the next section, '5.5 Implementing Drop-down Menus', will handle the selection of an index in which to write that value.

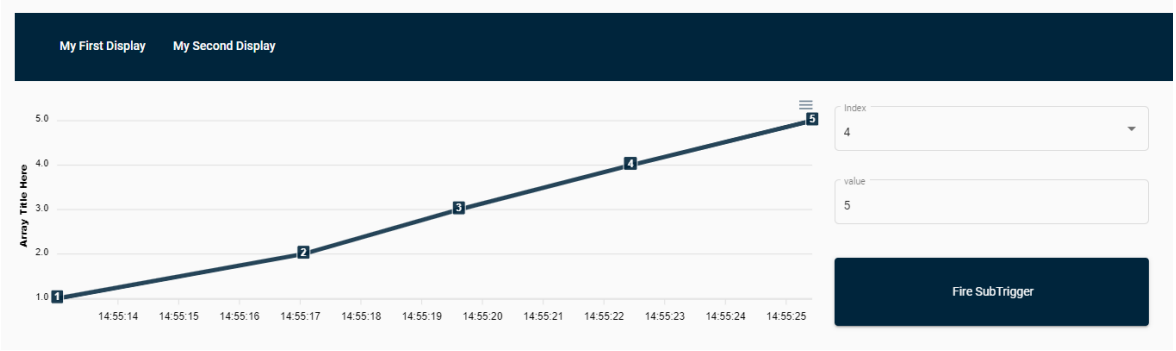
5.5. Implementing Drop-down Menus

5.5.1. Drag a 'Dropdown' widget onto the 'My Second Display' display above your 'Write Value' widget. In the widget configuration panel, add a new variable and assign the drop-down widget to write to the 'Index' local variable. Under 'Dropdown', ensure the 'Options Type' is set to 'Static Options' and enter "Index" for the 'Label'. Use the 'Options' menu to add five (5) different options, one for each index of the array. The 'Name' and 'Value' should match. These options will start at zero (0) rather than one (1) because we are dealing with an array.





5.5.2. Save and enter the live display. In order to update the line chart to show user inputted data over time, select index zero (0) from the drop-down widget menu and input any value into the 'Write Value' widget. Once both fields are populated, click on the 'Fire SubTrigger' button to fire the 'Update_Array' subtrigger while passing those values. Do this for all five (5) of the indexes while trying to input a different value for each. After all five (5) indexes have been populated with data, the line chart should show the user inputted values along with a time and date. Enable 'Data Labels' within the line chart widget configuration to show the integer values at each point without having to hover over the chart.



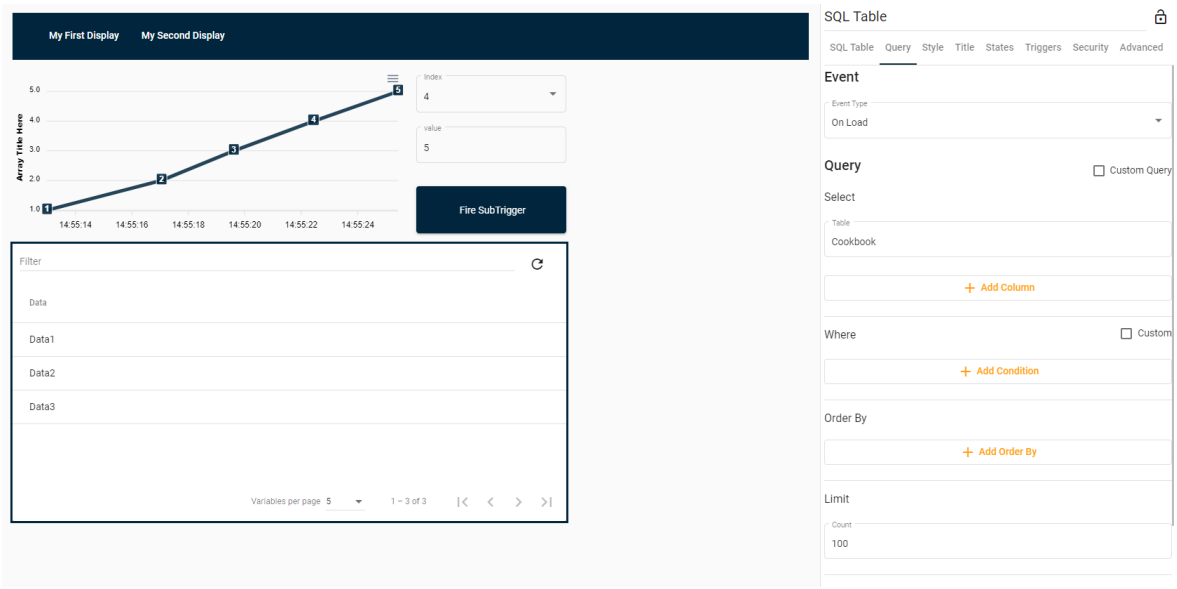
Cookbook / Devices

Devices | Variables | Data Mapping | Variable Groups

Name	Type	Value
Cookbook	Global Variables	
BOOL	BOOL	
FLOAT	FLOAT4	
INTEGER	INT4	
STRING	STRING(64)	
ARRAY	INT8[5,2]	
[0,0]	INT8	1642431313367
[0,1]	INT8	1
[1,0]	INT8	1642431317419
[1,1]	INT8	2
[2,0]	INT8	1642431319976
[2,1]	INT8	3
[3,0]	INT8	1642431322808
[3,1]	INT8	4
[4,0]	INT8	1642431325795
[4,1]	INT8	5

5.6. Viewing Database Content

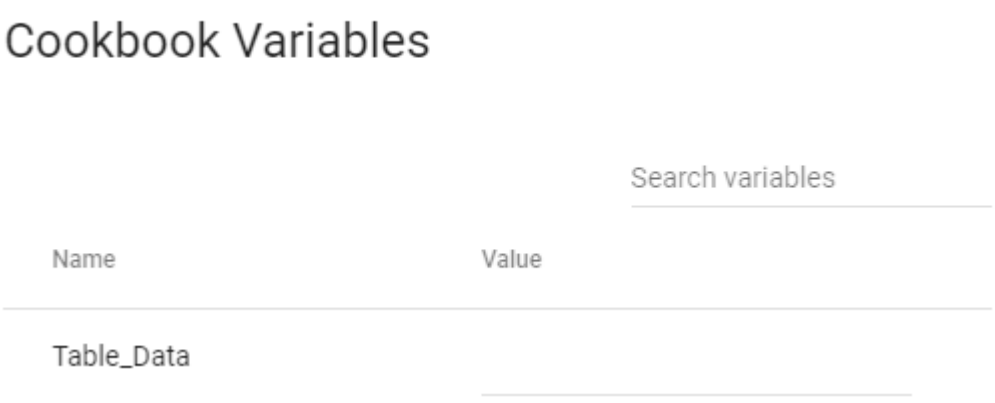
5.6.1. In deviceWISE VIEW, drag a ‘SQL Table’ widget onto ‘My Second Display’ below the trend chart. Enter the widget configuration and navigate to the ‘Query’ tab. Under ‘Select’, select the ‘Cookbook’ table created in ‘2.4. Creating a Local Database Table’. The SQL table widget will populate with the column and row data of the selected table. If it does not, enter and then exit live display VIEW or click the refresh button at the top right of the widget to force an update.



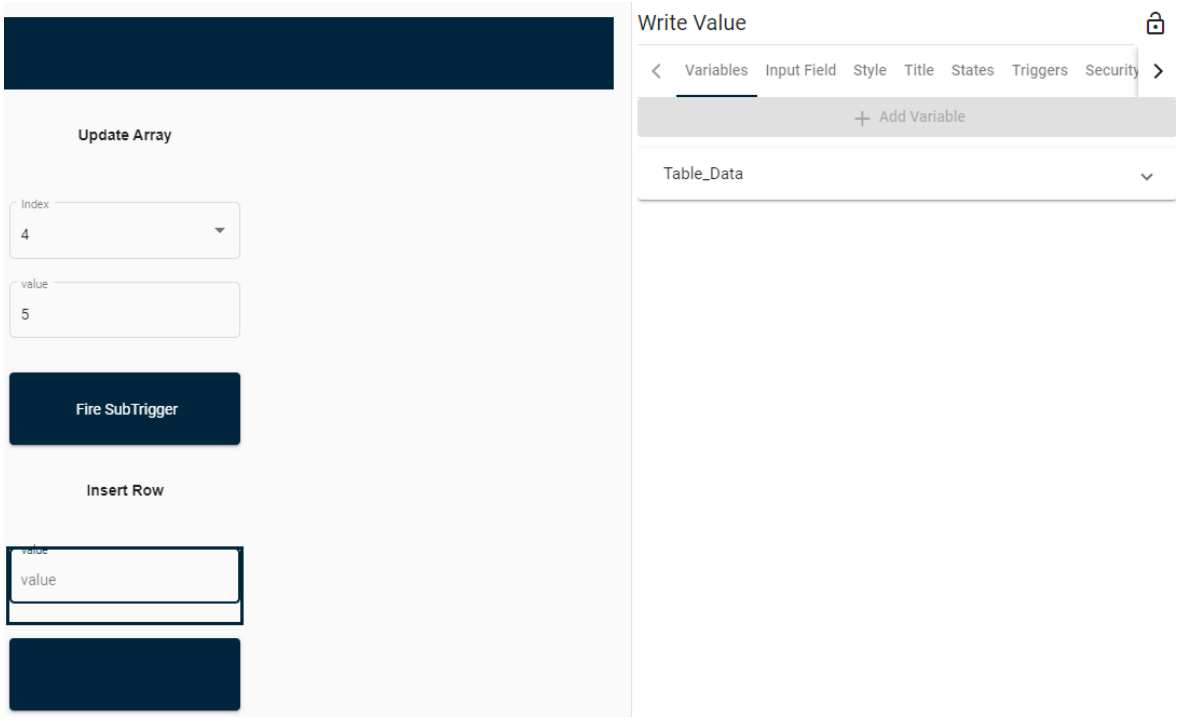
5.7. Inserting Database Content

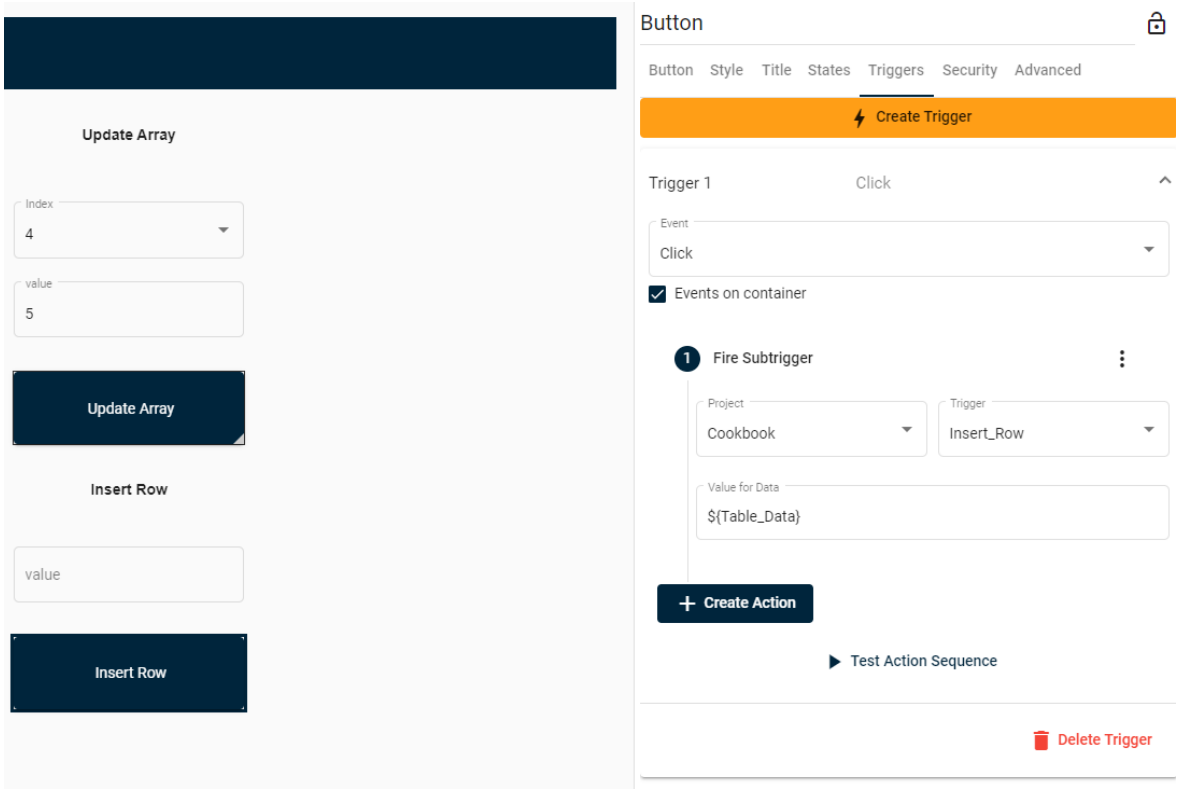
Utilizing the same methods demonstrated in ‘5.4. Calling SubTriggers with Parameters’, we can deploy our ‘Insert_Row’ subtrigger created in ‘2.5. Creating SubTriggers’ to insert user inputs directly into a local database table.

5.7.1. Create a deviceWISE VIEW local variable to handle this new input called “Table_Data” and ‘Save’.

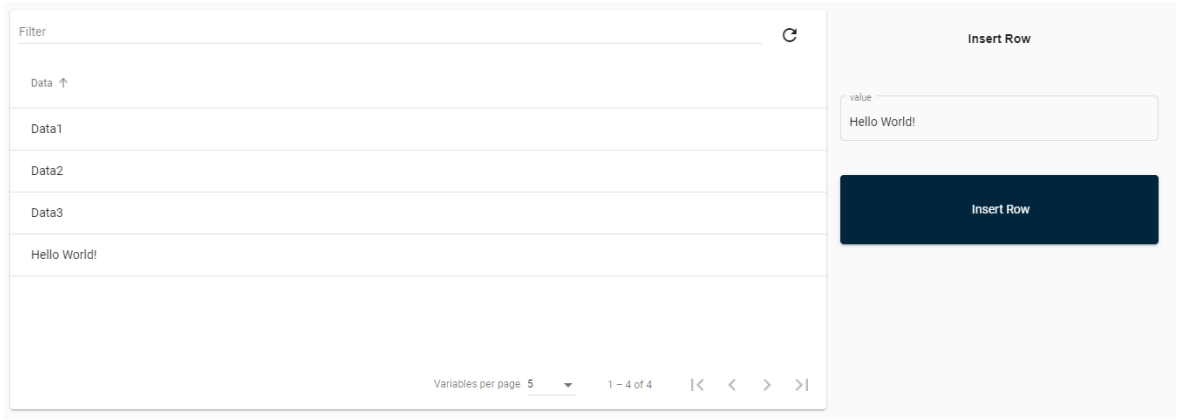


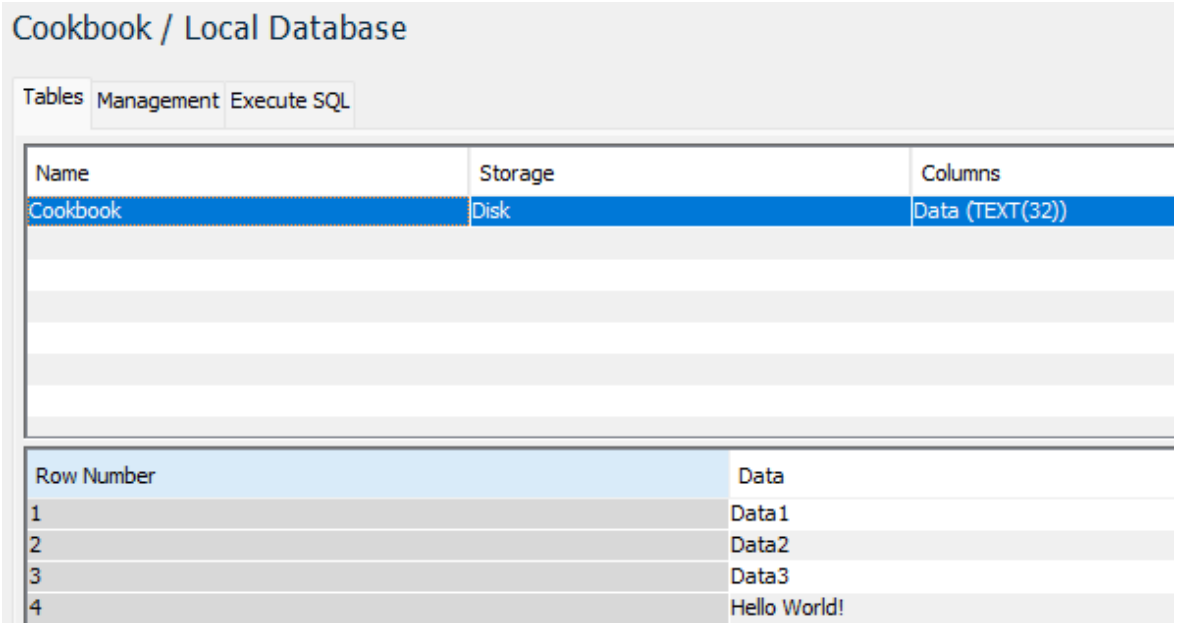
5.7.2. Drag a ‘Write Value’ and ‘Button’ widget onto your display. The user will input the desired row data into the ‘Write Value’ widget, which will then be passed to the ‘Table_Data’ local variable. The button will then fire our ‘Insert_Row’ subtrigger, pass the local variable, and insert its value directly into the ‘Cookbook’ database table. Configure the widgets as shown below. ‘Text’ widgets can also be deployed for organization.





5.7.3. Save and enter the live display. Input a value into the new 'Write Value' widget that you want inserted into our local database table and click on the button to fire the 'Insert_Row' subtrigger. After refreshing the 'SQL Table' widget by clicking the refresh icon at the top right of the widget, you should see the new value inserted in the bottom row. We can also check our 'Cookbook' database table on the Workbench to verify that the subtrigger executed correctly.



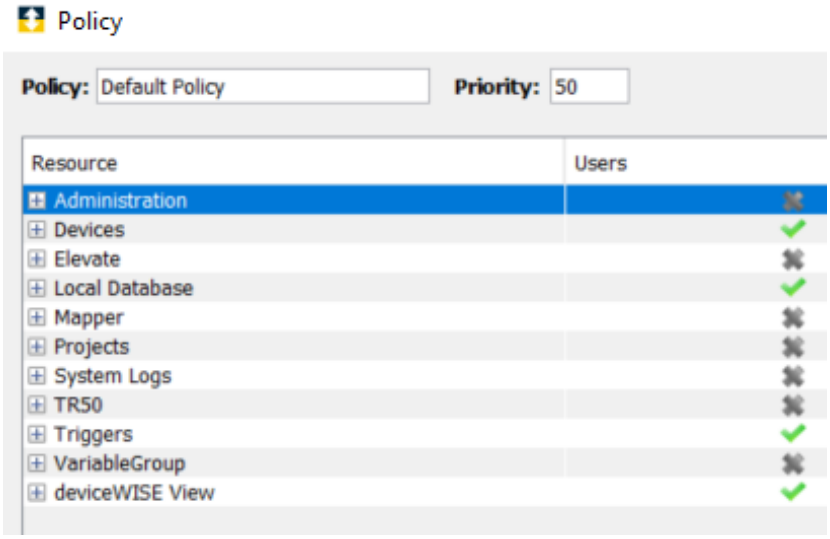


5.8. Implementing Security

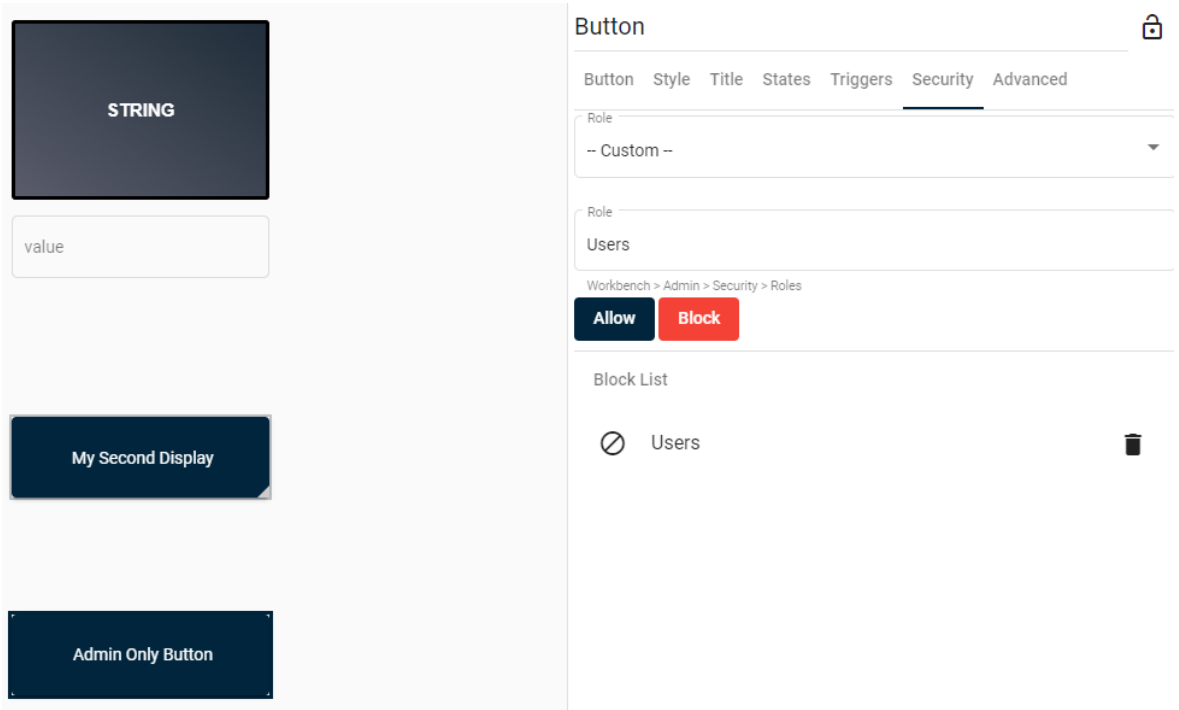
deviceWISE VIEW uses the built-in security features of deviceWISE to permit or prevent access to specific pages and widgets. When a widget is set to block a specific role, the widget will not be visible to any user in that role.

5.8.1. In the Workbench, select the 'Administration' dropdown from the left side panel of your node and locate the 'Security' tab. Here you can add new roles, users, and adjust the policies that each role must follow. By default, there is an 'admin' and 'user' role.

5.8.2. Navigate to the 'Policies' tab and double click the 'Default Policy' to begin adjusting the user policy settings. Ensure that 'Devices', 'Local Database', 'Triggers', and 'deviceWISE View' are checked. Click 'Save' and 'Yes' to apply.



5.8.3. On deviceWISE VIEW, drag a new 'Button' widget onto the 'My First Display' display. Name the button 'Admin Only Button' as we are going to specify security settings that only allow administrators to interact with or view the button. In the widget configuration panel, navigate to the 'Security' tab and change the 'Role' to '- Custom -'. Input 'Users' into the 'Role' field and click 'Block'. This will block normal users from interacting with the button or even being able to see it.



5.8.4. Log out of deviceWISE VIEW by clicking the user icon in the top right and selecting ‘Log Out’. Save any changes if prompted to do so. Log back into deviceWISE VIEW using the username ‘user’ and password ‘user’ to log in as a normal user. These credentials can be changed in the ‘Users’ section of the Workbench ‘Security’ tab when editing a user. In the ‘My First Display’ display, the ‘Admin Only Button’ will no longer be visible to normal users, while remaining visible when logged in as an administrator.

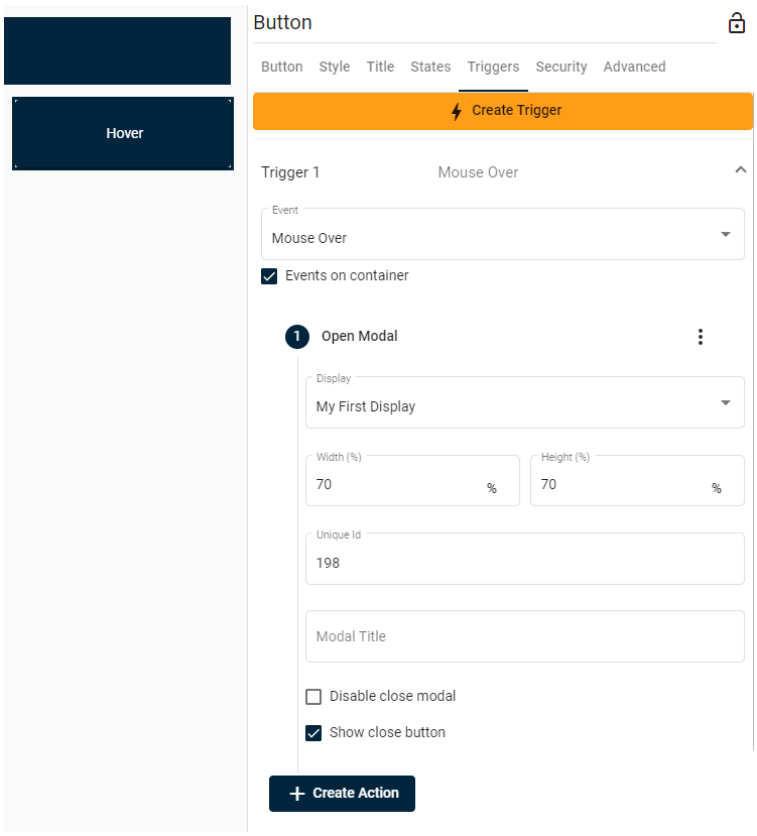
5.8.5. Every widget includes security settings that can prevent specific users from navigating to a certain page or viewing specific widgets. This can allow for deployments requiring administrator permissions to perform certain actions.

5.9. Implementing Pop-Up Displays and Text

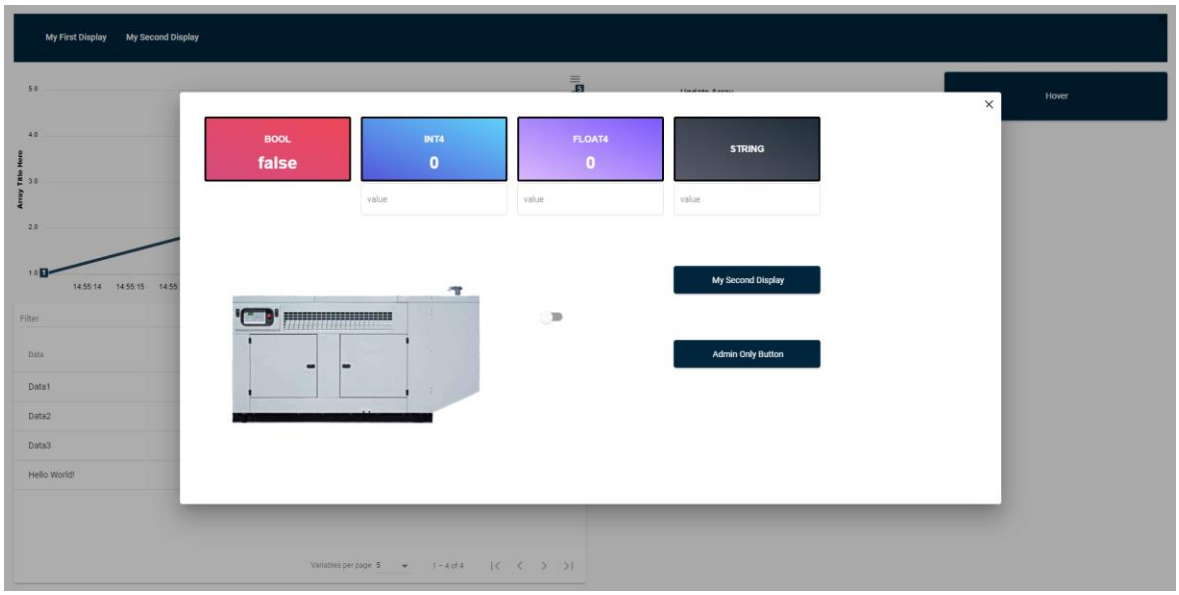
Within the widget configuration panel, there is a trigger action that allows a specified display to pop up on the screen without navigating to the display. This occurs when a certain event happens, such as clicking or hovering over a widget.

5.9.1. Drag another button widget onto ‘My Second Display’ and enter “Hover” for the displayed text. From the ‘Triggers’ tab, create a new trigger using the ‘Mouse Over’ event.

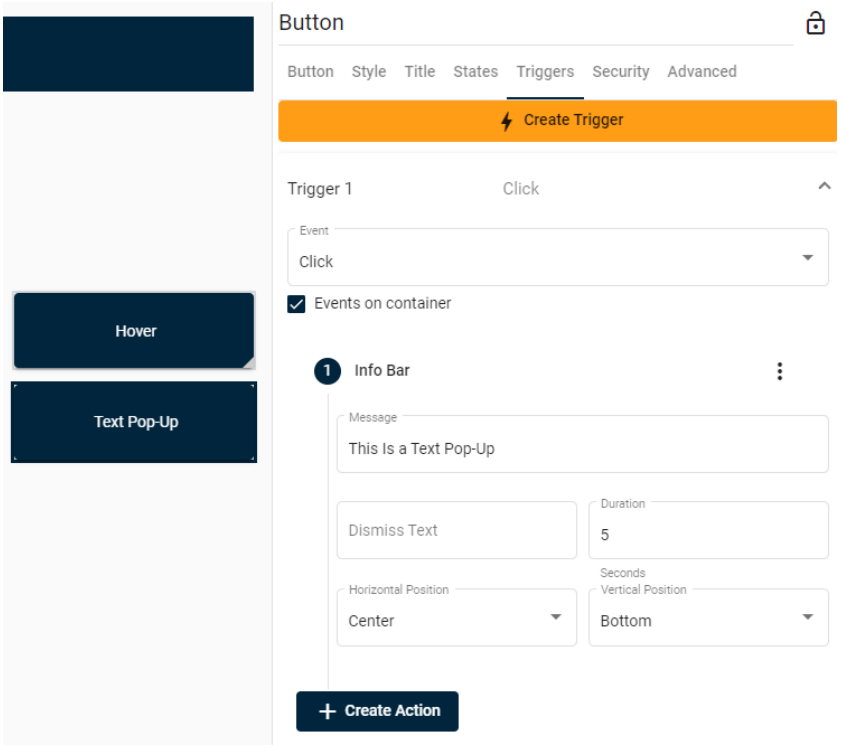
5.9.2. Create a new ‘Open Modal’ action and specify ‘My First Display’ as the display destination. Check ‘Show close button’. Save and enter the live display. Whenever a user hovers over the button, the ‘My First Display’ display will pop up on the screen, essentially “seeing” into the display without navigating to it. Click the close button or click outside of the modal to close it. Reference the images below.



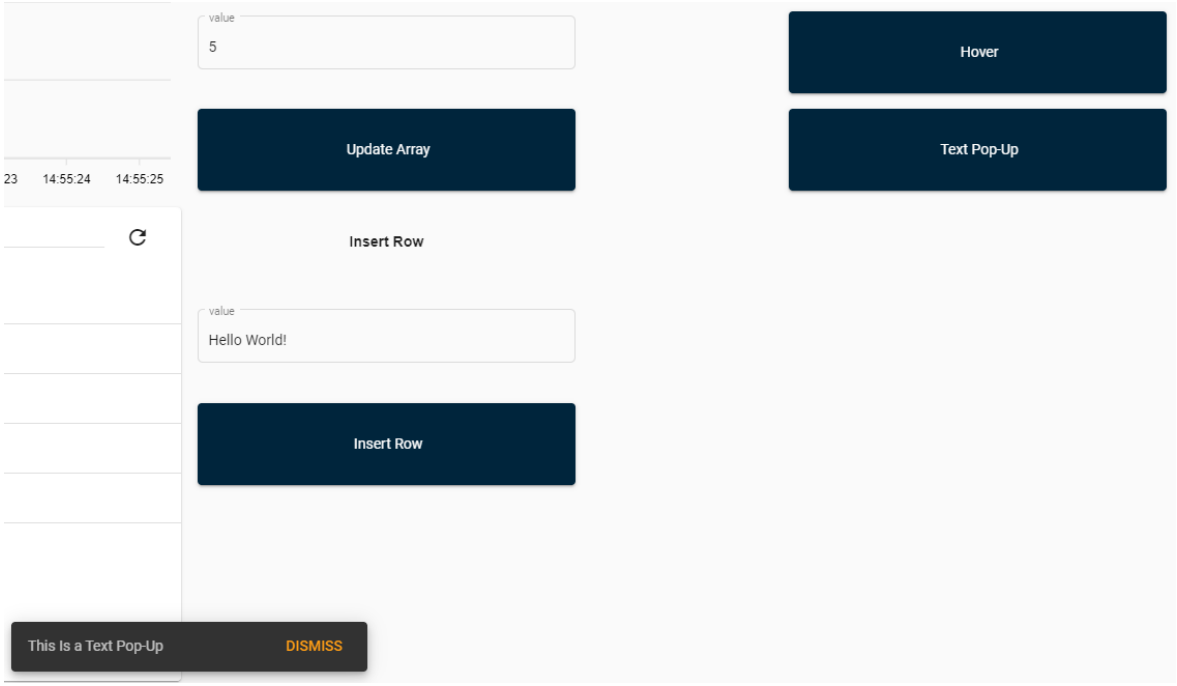
After Hovering



5.9.3. A text pop-up is performed in a similar fashion. Instead of an ‘Open Modal’ action, we will be using the ‘Info Bar’ trigger action. Create another button labeled “Text Pop-Up”. Within the ‘Triggers’ section of the widget configuration, add a click event that performs an ‘Info Bar’ action. Enter the message that will pop up on the screen and save. Enter the live display and click on the ‘Text Pop-Up’ button to trigger the pop-up text. Reference the screenshots below.



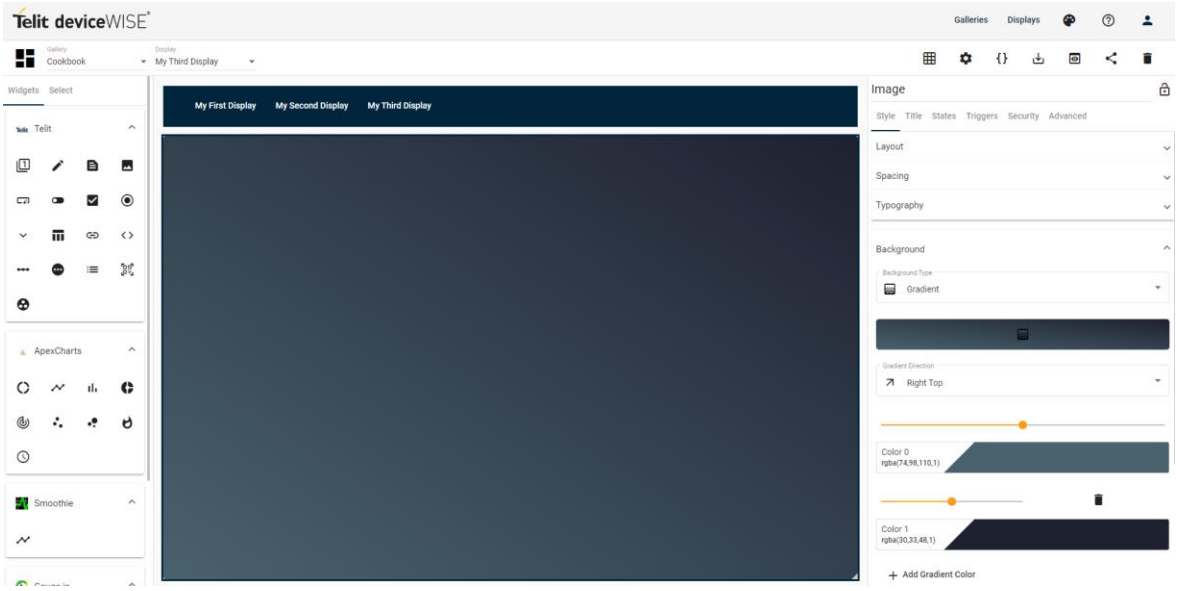
After Clicking



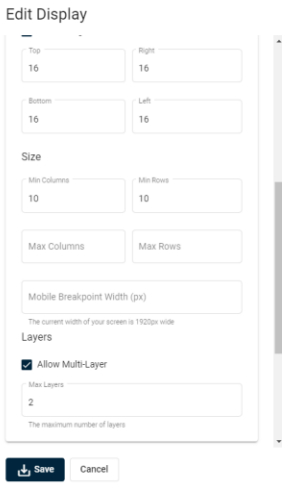
5.10. Using Layers

Layers are a useful tool that allows a user to place widgets on top or behind each other, rather than each widget having its own designated space within the display grid.

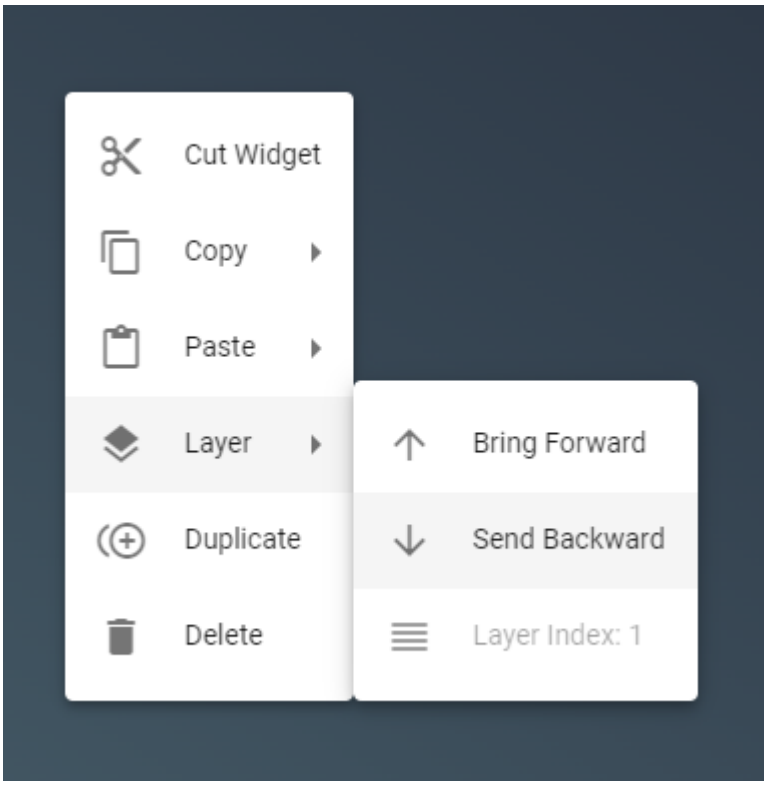
5.10.1. Create another display titled “My Third Display”. Add a navigation widget to the top of the display. Drag an ‘Image’ widget onto the display and resize it so it covers the remainder of the screen. Navigate to the ‘Style’ tab of the widget configuration and set a background under ‘Background’. For this example, I will set the background to a ‘Sleek’ gradient.



5.10.2. Click the cog wheel at the top right of the display labeled ‘Display Settings’ to allow layering. Under ‘Grid’, check ‘Allow Multi-Layer’. Set the max layers to two (2). Press ‘Save’.



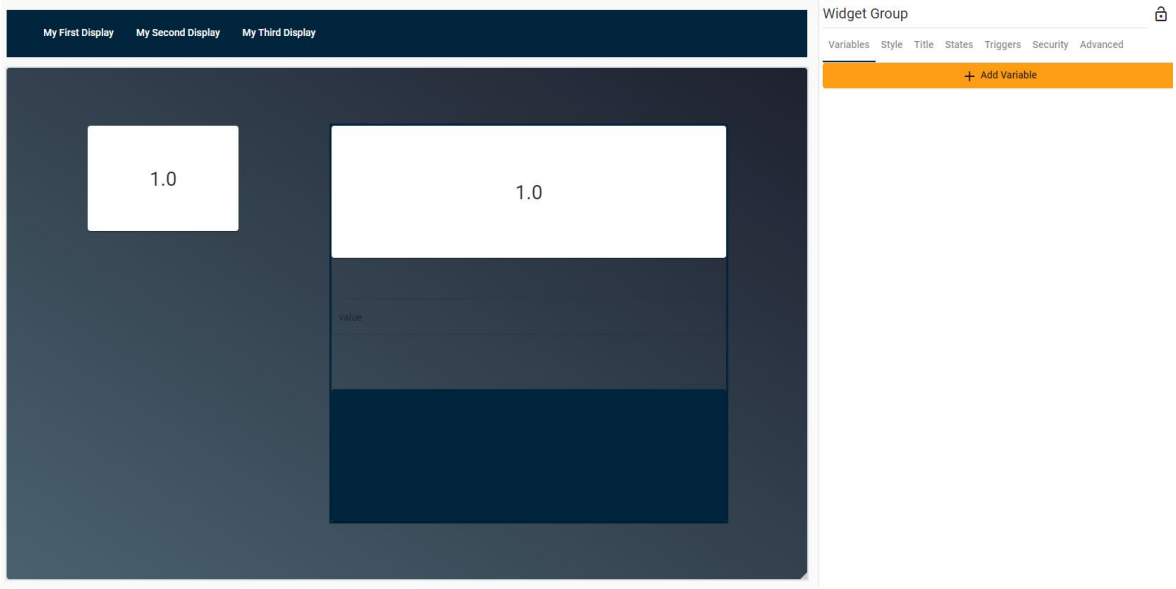
5.10.3. Right click the image widget and select ‘Layer’ -> ‘Send Backward’. Click and drag any widget onto the display, and you will notice that widgets are now able to be placed on top of the background image widget. Layering is available for every widget in deviceWISE VIEW.



5.11. Using Widget Groups

Several widgets can be inserted into a widget group in order to move, apply security settings, or perform any another configuration to a group of widgets at a time rather than each one individually.

5.11.1. In the ‘My Third Display’ display, drag a ‘Widget Group’ widget onto the workspace. Next, drag several widgets over the ‘Widget Group’ widget to insert them into the group.

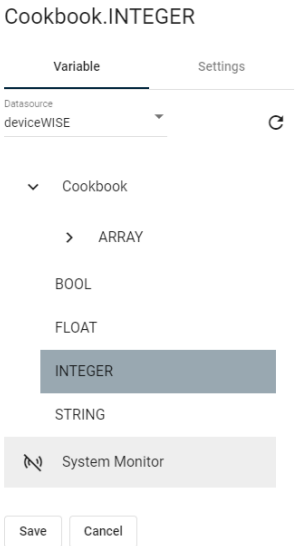


5.11.2. Any security configuration performed on the widget group applies to the entire group of widgets. You can also add background colors and utilize other styling options.

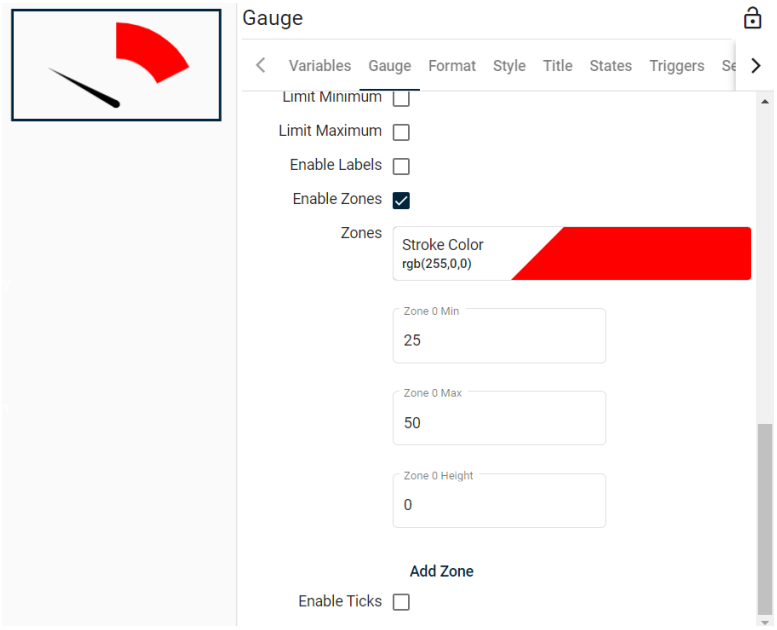
5.12. Using Gauges

deviceWISE VIEW has a collection of configurable gauge widgets that allow a user to completely customize the minimum and maximum values, create zones that correspond to color codes, and apply labels at specified values.

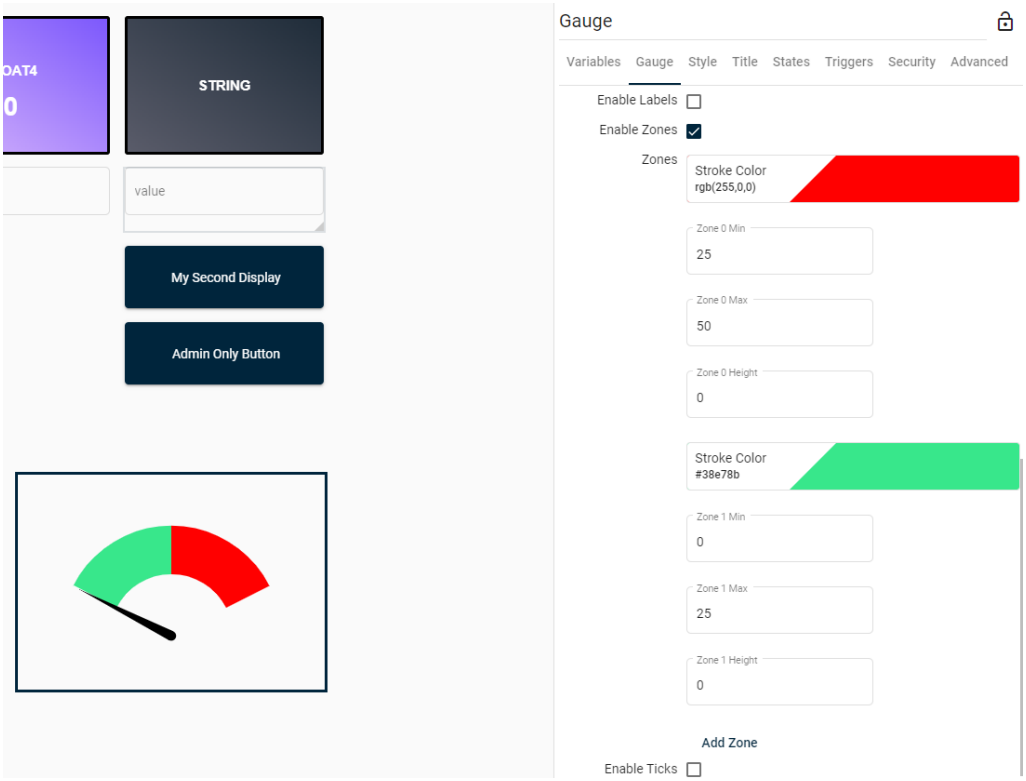
5.12.1. Navigate to the ‘My First Display’ display. From under the ‘Gauge.js’ widget section, drag a ‘Gauge’ widget onto the display. Specify the ‘INTEGER’ variable from the ‘Cookbook’ device by clicking ‘Add Variable’ from within the gauge widget configuration panel.



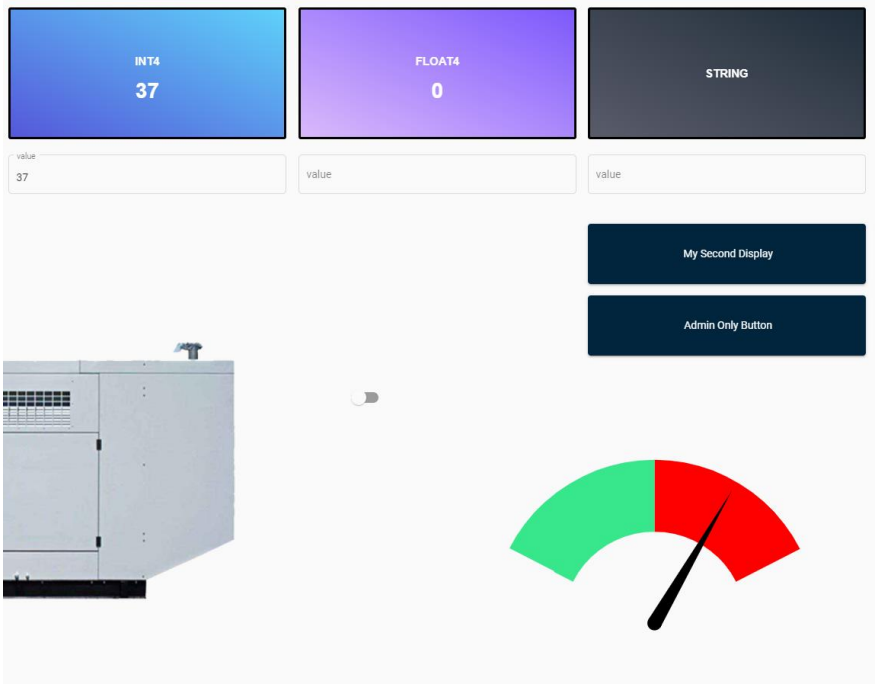
5.12.2. Under the ‘Gauge’ tab in the widget configuration, set the ‘Minimum’ value to 0 and the ‘Maximum’ value to 50. Scroll down, check ‘Enable Zones’, and add a zone. Set the ‘Zone Min’ to 25 and the ‘Zone Max’ to 50. Leave the ‘Stroke Color’ as the default red.



5.12.3. Add another zone by clicking ‘Add Zone’. This time, set the ‘Zone Min’ to 0 and the ‘Zone Max’ to 25. Set the zone’s ‘Stroke Color’ to a shade of green. For the sake of this guide, use #38e78b as the ‘Stroke Color’.



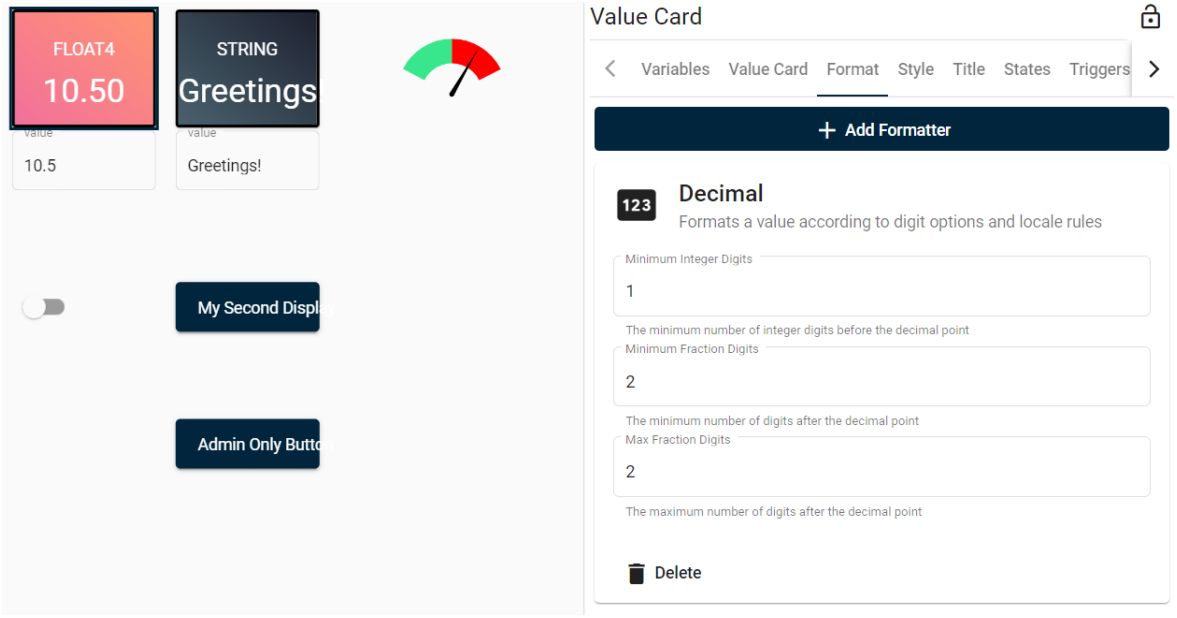
5.12.4. Save and enter the live display. Write to the 'INTEGER' variable using the write value widget created in '4.3. Writing Data in deviceWISE VIEW'. If you write a number greater than or equal to 25 and less than 50, you will notice the gauge pointer enter the red zone.



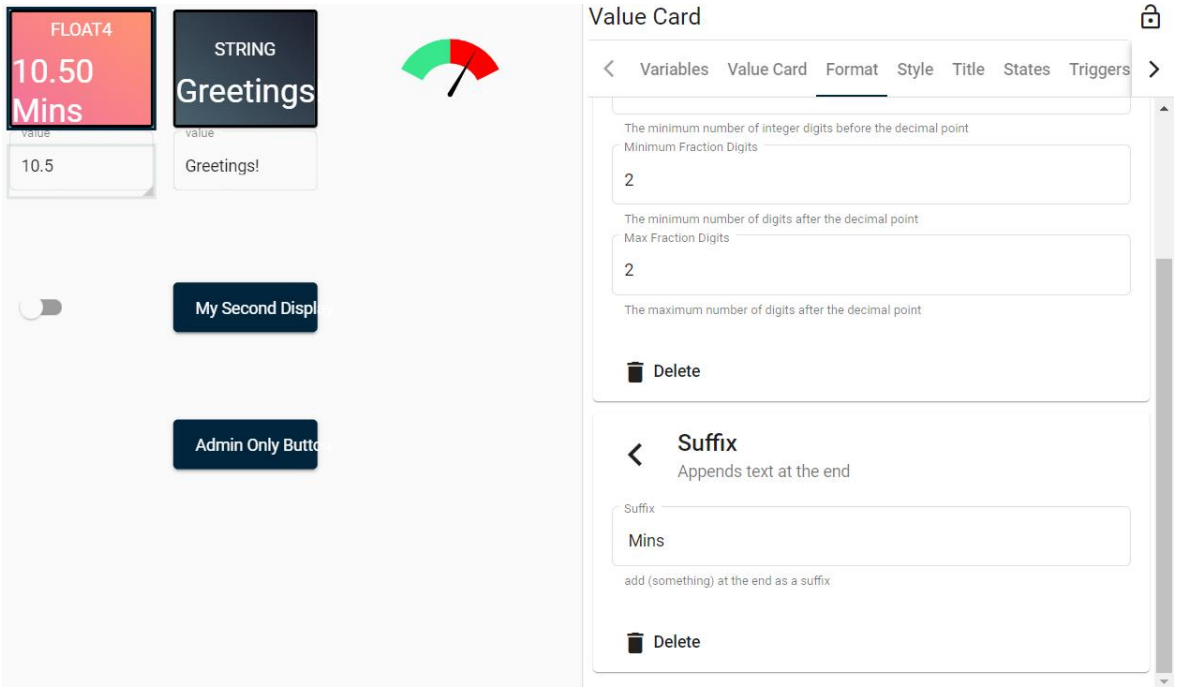
5.13. Widget Formatting

Every widget available within deviceWISE VIEW has a suite of formatting options allowing a user to add prefixes and suffixes to variable values, specify decimal places, and even perform math at the display level.

5.13.1. Navigate to the 'My First Display' display. Enter the widget configuration for the 'FLOAT4' value card. From within the 'Format' tab, add a new 'Decimal' formatter by clicking 'Add Formatter'. You will notice that the default decimal settings now allow 2 decimal places to be displayed on the value card.



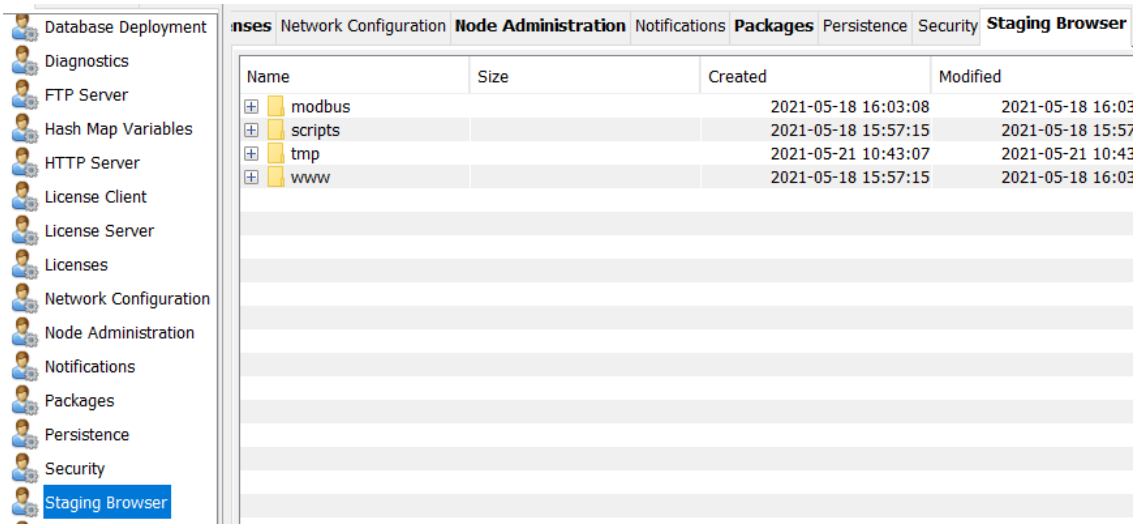
5.13.2. Add another ‘Suffix’ formatter. This applies a suffix after the value which is useful for specifying the unit of the value being displayed. For this example, enter “ Mins” as the suffix.



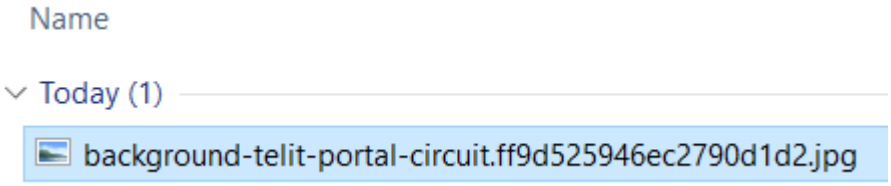
5.14. Customizing the Login Page

Within deviceWISE VIEW, you have the ability to customize nearly everything your user will have access to, including the login page. If you would like to change the login screen from the default Telit background, proceed with the steps below.

5.14.1. In the Workbench, select the 'Administration' dropdown from the left side panel of your node and navigate to the 'Staging Browser'.



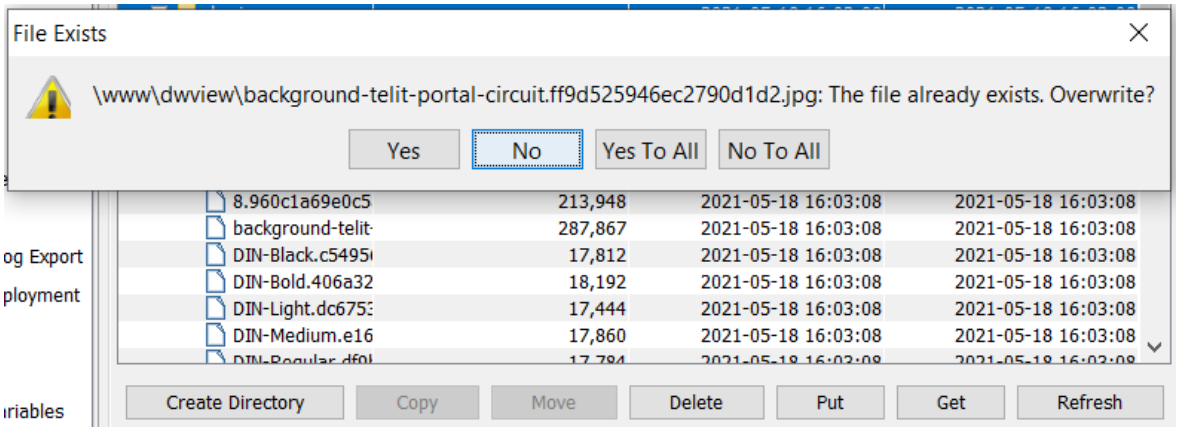
5.14.2. Navigate to the 'www' directory and open the folder titled 'dwview'. Within this folder, right click on the file containing 'background-telit-portal-circuit' and select 'Get' to download the file.



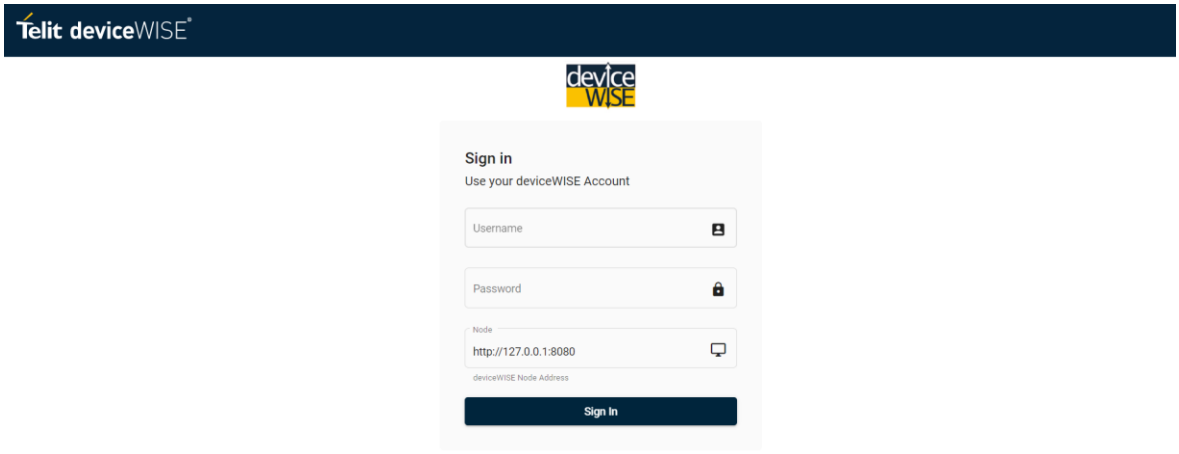
5.14.3. To replace the background of the default login screen, use any 1920x1080 .jpg image and rename it to exactly the file you downloaded in the previous step. You may overwrite this file to allow for easy renaming. For the sake of this guide, I will be using the image below.

If you would like to return to the default login page, create a copy of original .jpg downloaded from the Staging Browser to revert back to at any time.

5.14.4. After you are satisfied with an image and have overwritten or renamed it to the JPEG file downloaded earlier, navigate back to the ‘dwview’ folder within the ‘Staging Browser’. Upload your new ‘telit-background-portal’ JPEG file by right clicking the ‘dwview’ folder, selecting ‘Put’, and uploading your customized JPEG. Before uploading, the Workbench will ask if you would like to overwrite the file already located in the folder. Click ‘Yes’.



5.14.5. View your new login page by logging out of deviceWISE VIEW. If you do not see your new login page, close your web browser and open deviceWISE VIEW again.

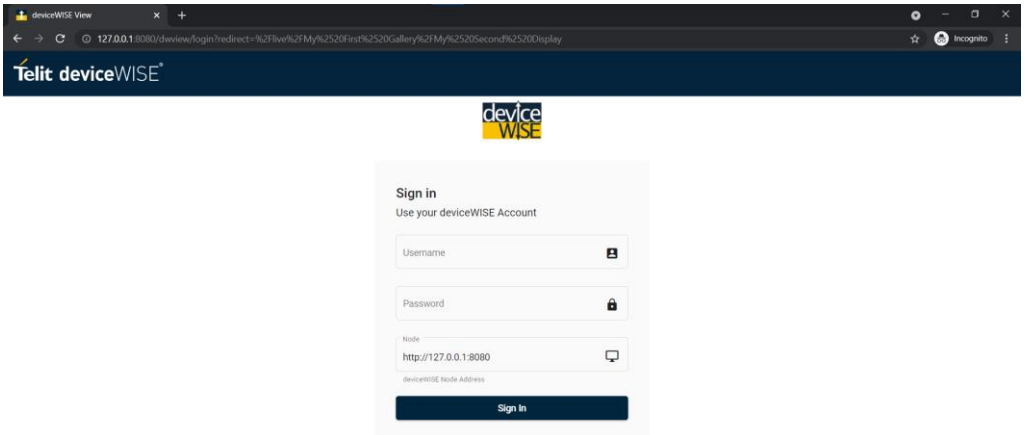


5.15. Extracting Display Links

When deploying your displays, it is important that a user should be able to access the screens without having to enter edit mode first. To do this, you must enter a live display and copy the URL of your web browser. When using the live display URL, it will prompt the user to log in and take them directly to the live display, bypassing the edit display screen. Sign out of deviceWISE VIEW first or open a new incognito tab to test.

Copied URL

After Pasting in New Incognito Window



5.16. Migrating to a Newer Version of VIEW

5.16.1.

When updating deviceWISE VIEW to a newer version, any galleries that were previously on the system will be considered out of date and may be incompatible. However, when entering an outdated gallery, VIEW will prompt the user to update the displays with the click of a button. Click ‘Update Gallery’ to automatically migrate any outdated displays to the newest installed version of deviceWISE VIEW.

6. GLOSSARY AND ACRONYMS

	Description
KPI	Key Performance Indicator
HMI	Human-Machine Interface
Variable	Container with an associated symbolic name that stores different types of data
BOOL	Data type that has two possible values representing truth in logic
INT	Data type that stores a number value written without a fractional component
FLOAT	Data type that stores double-precision floating-point numbers with up to 17 significant digits
STRING	Data type the stores a sequence of characters
Array	Data structure consisting of a collection of values each identified by at least one array index or key
Database	Organized collection of data, generally stored and accessed from a computer system
SubTrigger	A trigger event type that is executed when called upon by another trigger or deviceWISE VIEW, allowing for the passing of parameters.
IP	Internet Protocol
IP Address	Unique address that identifies a device on the internet or a local network
HTTP	Hypertext Transfer Protocol
URL	Uniform Resource Locator
Gallery	Folder that contains a collection of displays
PLC	Programmable Logic Controller
X-Axis	Horizontal axis of a system of coordinates
Y-Axis	Vertical axis of a system of coordinates
SQL	Structured Query Language

7. DOCUMENT HISTORY

Revision	Date	Changes
1	2021-05-18	Initial Draft
2	2021-06-17	deviceWISE VIEW Migration Tool
3	2021-08-04	21.07 Update, Security Features
4	2021-08-12	Modals, Layering
5	2021-09-09	deviceWISE VIEW 1.2.1, Drop-down Options, Gauge Widget, Formatting
6	2022-01-17	deviceWISE VIEW 1.3.0, Trend Charts, Widget Grouping

SUPPORT INQUIRIES

Link to **www.telit.com** and contact our technical support team for any questions related to technical issues.

www.telit.com

Telit Communications S.p.A.
Via Stazione di Prosecco, 5/B
I-34010 Sgonico (Trieste), Italy

Telit IoT Platforms LLC
5300 Broken Sound Blvd, Suite 150
Boca Raton, FL 33487, USA

Telit Wireless Solutions Inc.
3131 RDU Center Drive, Suite 135
Morrisville, NC 27560, USA

Telit Wireless Solutions Co., Ltd.
8th Fl., Shinyoung Securities Bld.
6, Gukjegeumyung-ro8-gil, Yeongdeungpo-gu
Seoul, 150-884, Korea

Telit Wireless Solutions Ltd.
10 Habarzel St.
Tel Aviv 69710, Israel

Telit Wireless Solutions
Technologia e Servicos Ltda
Avenida Paulista, 1776, Room 10.C
01310-921 São Paulo, Brazil

Mod. 2022-01 Rev.6

01.2022