Px Compiler

User Guide and Reference

ddcToolbox.com

Rev. September 10, 2021

Table of Contents

Overview

Px Compiler is a utility program that adds enhanced graphic editing features to Alerton's Compass software. One of the main features allows for embedding graphic displays within another graphic display. This simple, yet powerful, feature makes it possible to "nest" a display (or template) onto many other displays without relying on copy/paste. This makes the process of creating many displays (that share common elements) extremely fast and consistent. Changes to a nested display will also apply the changes to all graphics that contain the nested display. This feature when combined with other features of Px Compiler are extremely useful when creating the following:

- Equipment "quick view" graphics
- "floor plan" graphics
- Navigation links (shared across many displays)
- headers (shared across many displays)
- footers (shared across many displays)
- "floor plan" overview maps (shared across multiple displays)

Variable injection is another powerful feature of Px Compiler that aids in the development of graphics. Data contained in the Compass Device Manager table can now be used directly in a graphic (at compile time, aka when the display is saved). Px Compiler also allows for viewing and editing the Compass Device Manager table directly. The Px Compiler has an import and an export feature which make adding new devices, bulk editing, sorting, and filtering device data simple.

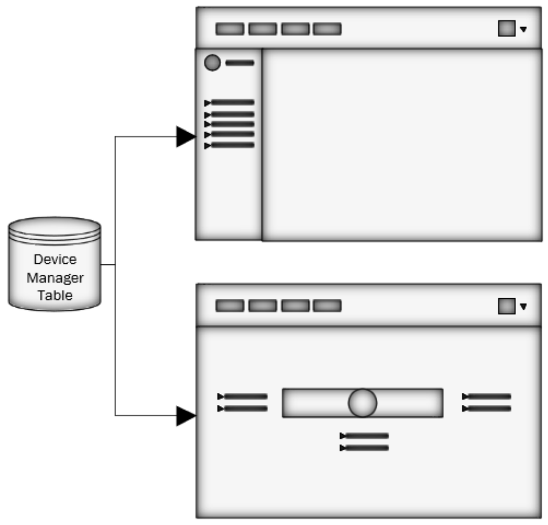


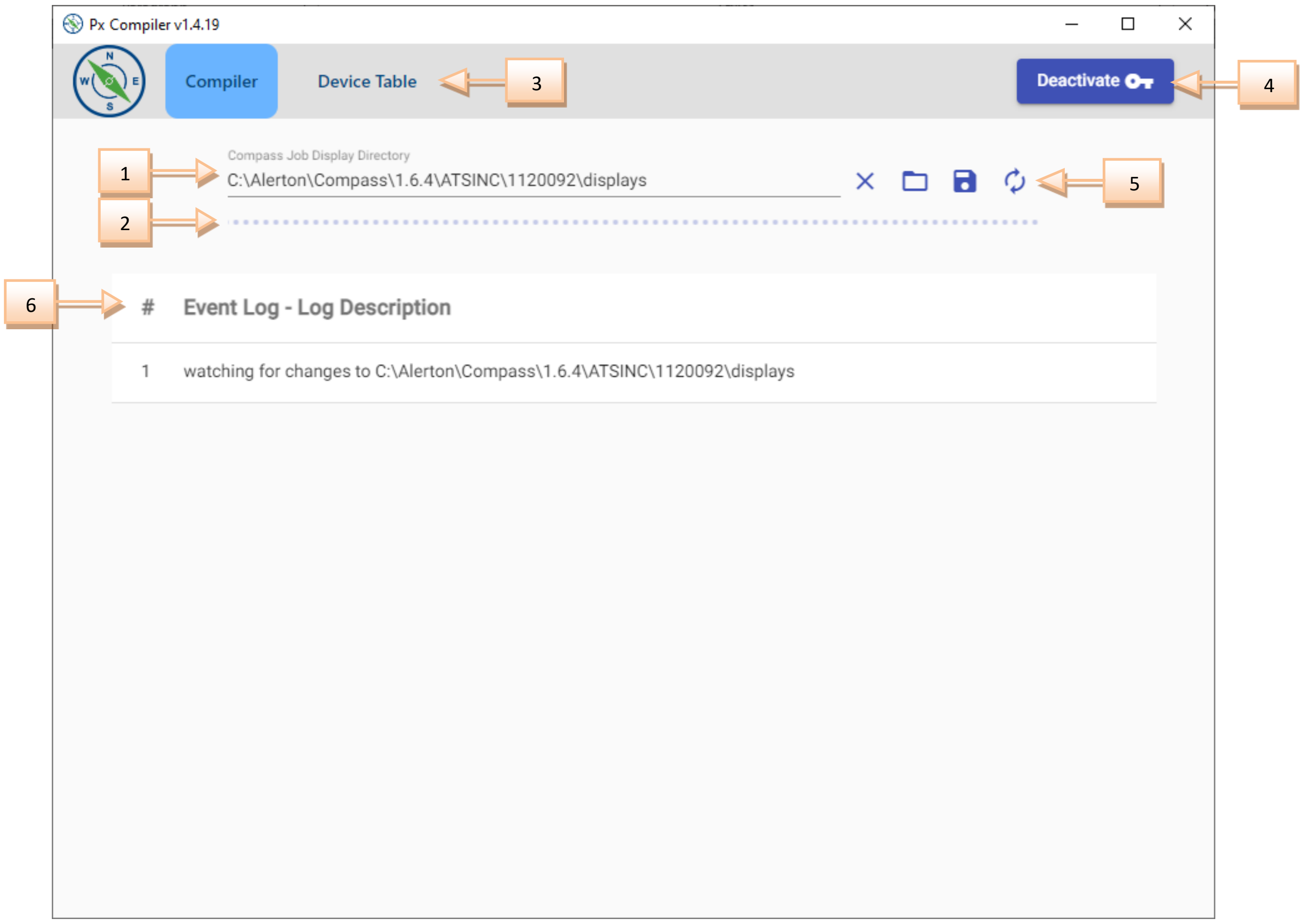
Figure 1 - Flow of data from Device Manager to Graphics

How it works…

Px Compiler works by running in the background on a PC that is also running Alerton Compass software. When the graphics (displays and templates) are edited and saved in Compass, Px Compiler will automatically “go to work” on them. Px Compiler will make the necessary changes based on special commands that may be present in the graphics file. All graphical work is done completely in the Compass IDE which makes it easy to use Px Compiler. Aside from telling Px Compiler which Alerton Rep/Job to monitor for changes, the user does not need to interact with Px Compiler in any way to utilize most graphical features. The Graphics Development section explains, at a high level, how to use this program to develop graphics. Refer to the Syntax and Commands section which lists all available commands and explains how to build expressions within Compass graphics.

User Interface

Refer to Figure 2 & Figure 3 for an overview and explanation of each element of the user interface. Most elements, when hovered over, will display a tooltip containing supplemental information.



C:\Alerton\Compass\1.6.4\ATSINC\1120092\displays

1 watching for changes to C:\Alerton\Compass\1.6.4\ATSINC\1120092\displays

Figure 2 - Compiler View

Compiler View

1. The **Directory** field contains the path to the current Compass REP/JOB “Displays” folder. On initial startup, a default value will be shown.
2. The **Status Bar** is displayed only when Px Compiler is actively watching for changes to the displays in the specified REP/JOB. Alternatively, a separate progress bar will be visible below the Status Bar when Px Compiler is actively analyzing display files or compiling display files.
3. The **Navigation Bar** is used to switch between the different screens of the program.

4. This **Activate/Deactivate** button brings up a separate window where users can activate or deactivate the software by entering a purchased activation code. This window will also display the current activation code and expiration date (if applicable).

5. The action buttons perform various tasks:

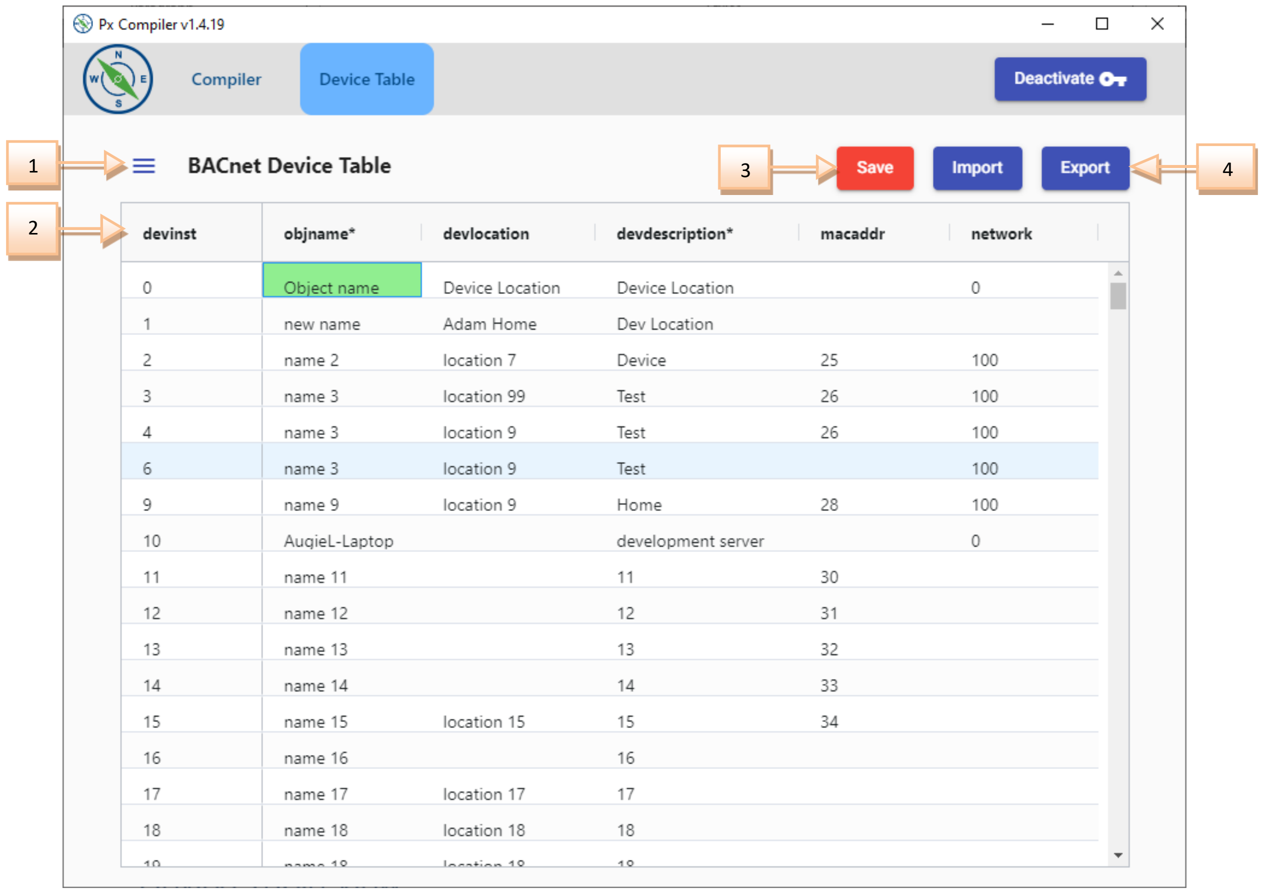
    a. **Stop** will stop the program from watching for display file changes. The Status Bar will then disappear.

    b. **Open** will open a folder dialog window which can be used to change the Compass REP/JOB display folder and make the program start watching for display file changes. The Status Bar will then appear.

    c. **Save/Convert** will begin the conversion operation which converts the .px graphics to an equivalent .DSPX or .DVTX file. This is used for "locking in" graphics so future edits to graphics do not require the use of this software.

    d. **Compile All** will parse through and compile all display and template files in the current Compass project.

6. The **Event Log** displays all events/operations that Px Compiler has performed during the current instance. This log is cleared when the program is closed.



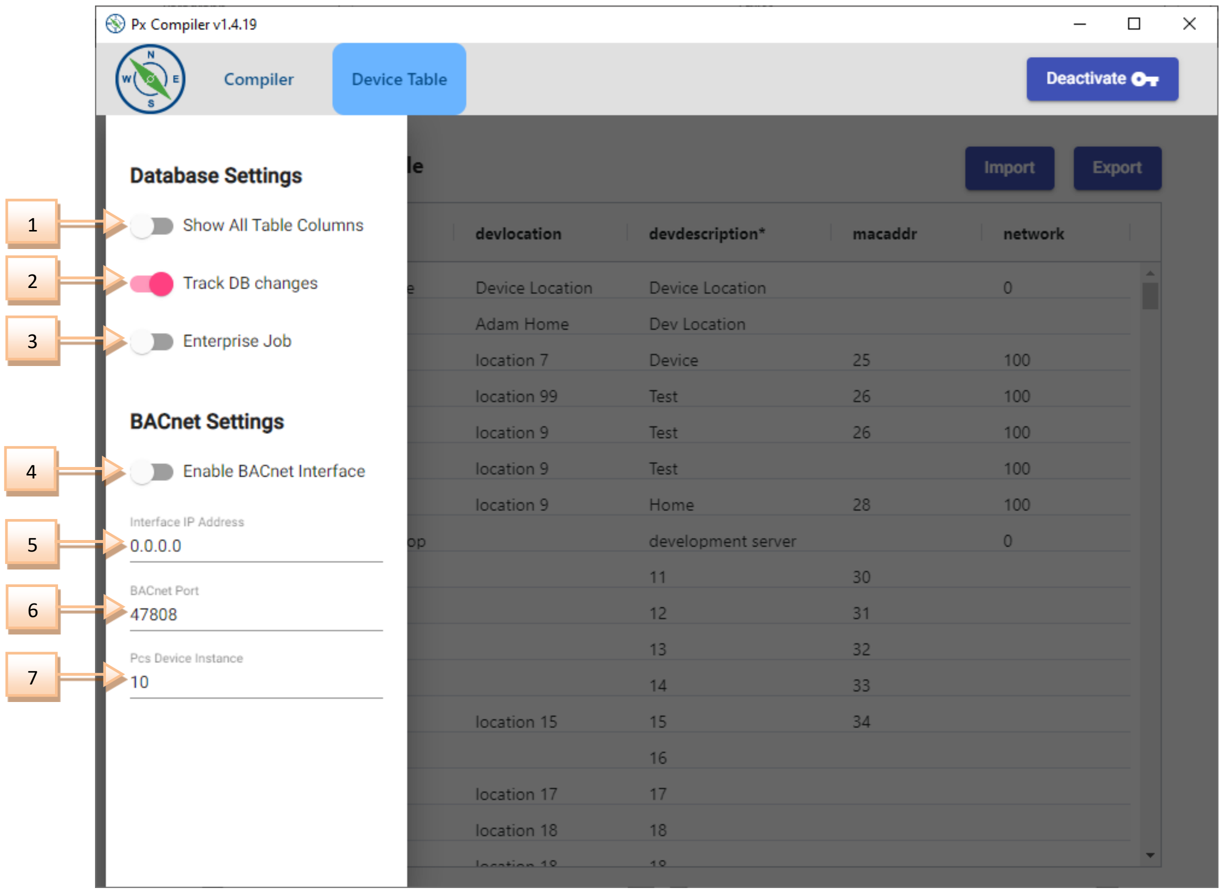
devinst	objname*	devlocation	devdescription*	macaddr	network
0	Object name	Device Location	Device Location		0
1	new name	Adam Home	Dev Location		
2	name 2	location 7	Device	25	100
3	name 3	location 99	Test	26	100
4	name 3	location 9	Test	26	100
6	name 3	location 9	Test		100
9	name 9	location 9	Home	28	100
10	AugieL-Laptop		development server		0
11	name 11		11	30	
12	name 12		12	31	
13	name 13		13	32	
14	name 14		14	33	
15	name 15	location 15	15	34	
16	name 16		16		
17	name 17	location 17	17		
18	name 18	location 18	18		

Figure 3 - Device Table View

Device Table View

1. The "hamburger" menu button will open the settings sidebar. Within this sidebar are various settings that users may adjust.
2. The **Device Table** contains every device within device table of the BACtalk.mdb or SQLEXPRESS database of the current REP/JOB. All cells (except for devinst) are editable. Any edited, and un-saved, cell will turn green to indicate that it has not yet been saved to the database. All columns can be filtered, sorted, resized, pinned, reordered, hidden, and reset.
3. The **Save** button is visible only when there are edited, and un-saved, cells in the Device Table. Clicking this button will save all changes to the database. All green cells will then change to white and the save button will disappear.
4. Data can be imported from Excel into the Device Manager table, or data from the Device Manager table can be exported to Excel.
    a. **Import** button will open the Import Settings dialog where the target Excel Workbook and Worksheet can be selected. See below for more information on this dialog window.

b. **Export** button will open the Export Settings dialog where the target Excel Workbook and Worksheet can be selected. See below for more information on this dialog window.



Database Settings

- [ ] Show All Table Columns
- [✓] Track DB changes
- [ ] Enterprise Job

BACnet Settings

- [ ] Enable BACnet Interface

Interface IP Address: 0.0.0.0

BACnet Port: 47808

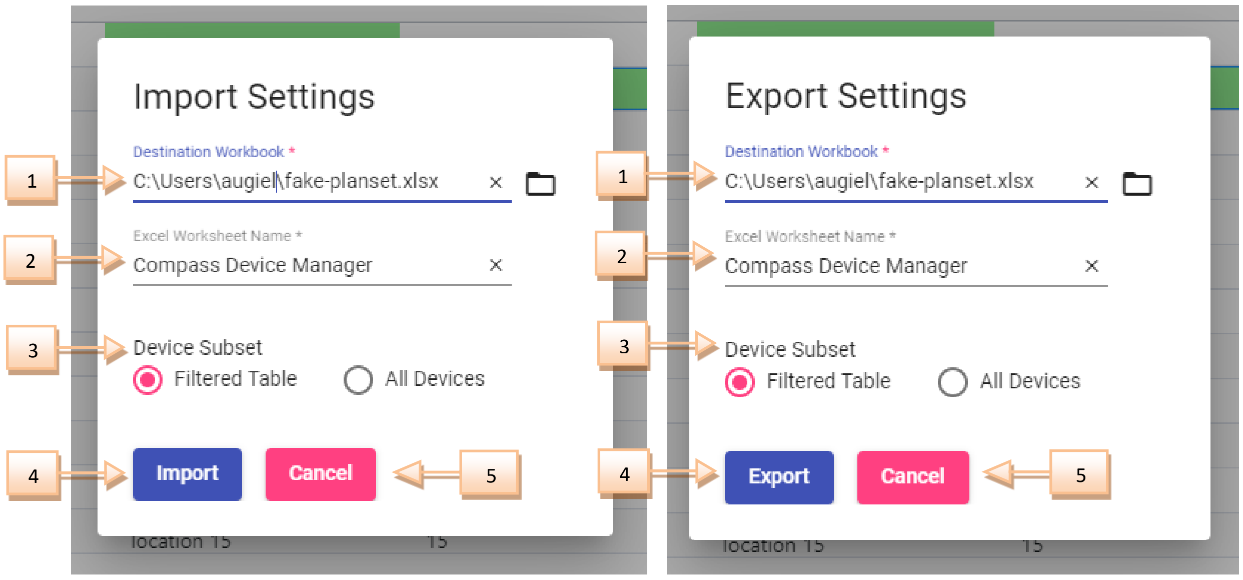
Pcs Device Instance: 10

	devlocation	devdescription*	macaddr	network
e	Device Location	Device Location		0
	Adam Home	Dev Location		
	location 7	Device	25	100
	location 99	Test	26	100
	location 9	Test	26	100
	location 9	Test		100
	location 9	Home	28	100
op		development server		0
		11	30	
		12	31	
		13	32	
		14	33	
	location 15	15	34	
		16		
	location 17	17		
	location 18	18		

Figure 4 - Settings Sidebar

Device Table - Settings Sidebar View

1. The **Show All Table Columns** toggle will show/hide the extra fields in the Device Table. When off, only the most common fields are visible.
2. The **Track DB Changes** toggle will enable/disable the Bactalk database tracking feature.


Import Settings

Destination Workbook *
C:\Users\augiel\fake-planset.xlsx

Excel Worksheet Name *
Compass Device Manager

Device Subset
[✓ Filtered Table] All Devices

Export Settings

Destination Workbook *
C:\Users\augiel\fake-planset.xlsx

Excel Worksheet Name *
Compass Device Manager

Device Subset
[✓ Filtered Table] All Devices

Figure 5 - Import/Export Dialog

Import Dialog

1. The **Destination Workbook** field contains the path to an existing Excel Workbook to import from. Edit this field by clicking on the **Folder** icon to open a file dialog, then select the target Workbook. OR copy/paste the path (including filename and extension) into the field.
2. The **Excel Worksheet Name** field is the name of the Excel Worksheet which will be imported. The names must match exactly (case-sensitive). The default value of **"Compass Device Manager"** can be changed by clicking in the field and typing a new value.
3. The **Device Subset** toggle determines whether all data will be imported from the Excel file, or only a subset of data will be imported.
    a. When **Filtered Table** is selected, only data for devices that are currently in the Device Table will be imported. Devices that are not in this table, will be ignored during the import. Filters currently applied to the Device Table will also be considered. I.e., A user applies a filter to only show devices with a device instance that starts with "100". After Importing, only the devices shown in the Device Manager table will be affected.
    b. When **All Devices** is selected, all devices will be imported. Devices that exist in the Excel Workbook but not in the Device Manager will be added and placed at the top of the table. Active table filters do not apply. This is useful for adding new devices to a project, in bulk.
4. Click **Import** to begin the import operation. If an error occurs, a popup displaying the error message will appear. Otherwise, all edited cells and new devices added will have green cells and the Save button will appear. This allows a user to preview the changes prior to saving to the

database. To discard all edits without saving, simply navigate to the Compiler View, then back to this view.

5. Click **Cancel** to cancel the import and exit the dialog window.

The format of the imported worksheet must be formatted properly, otherwise the import is likely to fail. Use the Export feature to obtain a properly formatted worksheet.

Export Dialog

1. The **Destination Workbook** field contains the path to an existing Excel Workbook to export to. Edit this field by clicking on the **Folder** icon to open a file dialog, then select the target Workbook. OR copy/paste the path (including filename and extension) into the field.
2. The **Excel Worksheet Name** field is the name of the Excel Worksheet which will be imported. This worksheet will be created, and a worksheet of this name MUST NOT already exist. Otherwise, the export operation will fail. The default value of **"Compass Device Manager"** can be changed by clicking in the field and typing a new value.
3. The **Device Subset** toggle determines whether all data will be exported to the Excel file, or only a subset of data will be exported.
    a. When **Filtered Table** is selected, only data for devices that are currently in the Device Table will be exported. Devices that are not in this table, will be ignored during the export. Filters currently applied to the Device Table will also be considered. I.e., A user applies a filter to only show devices with a device instance that starts with "100". After exporting, only the devices shown in the Device Manager table will be exported to the Excel file.
    b. When **All Devices** is selected, all devices will be exported. Active table filters do not apply. This is useful for exporting an entire projects device table.
4. Click **Export** to begin the export operation. If an error occurs, a popup displaying the error message will appear.
5. Click **Cancel** to cancel the export and exit the dialog window.

Initial Setup

Getting started using Px Compiler is easy. First, download and install the program by running the provided ".exe" installer file. After that, simply start the program and set Px Compilers path to the desired Compass REP/JOB Display folder. The program will start watching for changes to your displays and this program will automatically go to work as you build graphics directly in Compass.

The next time this program is opened, it will start "watching" the last folder that it was set to watch (not available in "Evaluation Mode".

The program will be in "**Evaluation mode"** until it has been activated with an activation code. However, many of the features are still available to use in this mode.

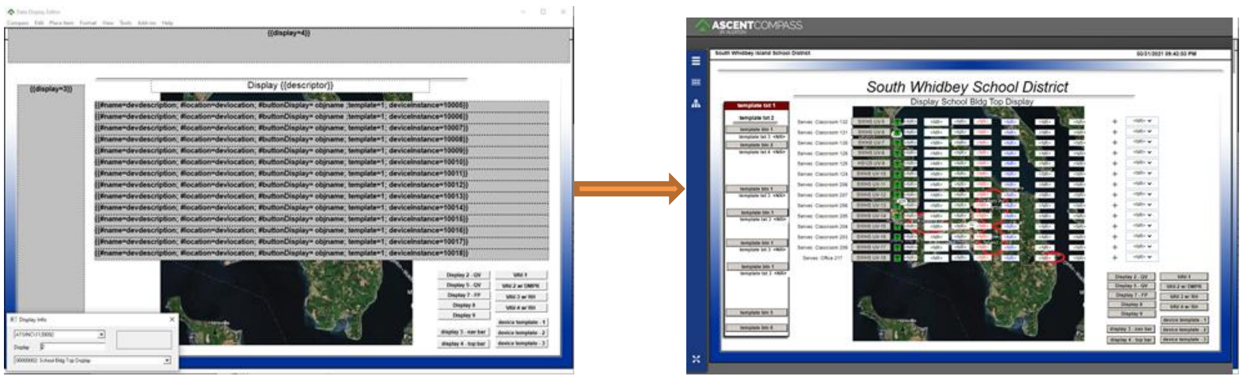
While in "Evaluation Mode", all displays that use features from this program will have watermarks added. The

The display "Conversion" feature is disabled in "Evaluation Mode".

Graphics Development

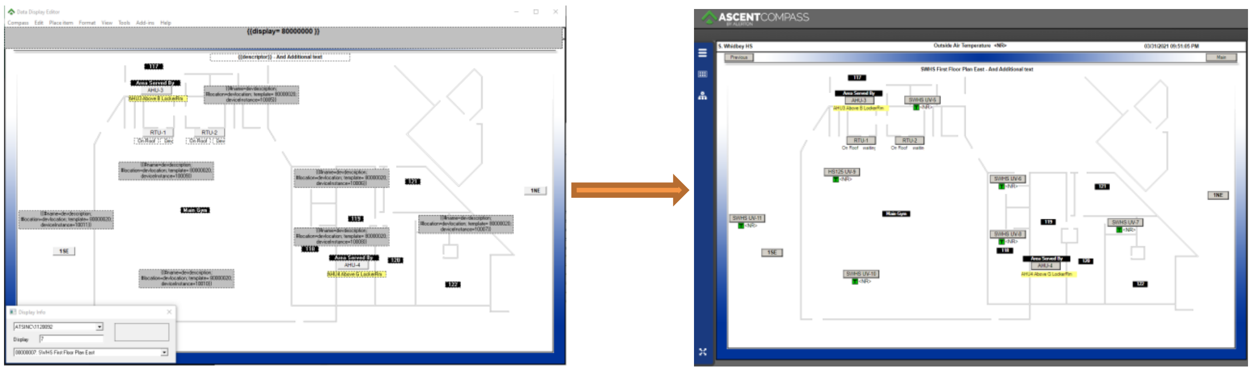
This section will provide guidance for using the features of this program to enhance the development of graphics. Refer to the Syntax and Commands section for specific instructions on how to use these features.

Figure 5 below demonstrates how to utilize templates to build a quick view graphic; while Figure 6 demonstrates how to utilize templates to build a floor plan graphic. The images on the left are what the Compass graphic windows looks like. The images on the right are the output of those displays in a browser (assume that Px Compiler is running on this machine and set to this REP/JOB).



```
{{display=4}}
{{display=3}}
Display {{descriptor}}
{{#name=devdescription; #location=devlocation; #buttonDisplay= objname ;template=1; deviceinstance=10005}}
{{#name=devdescription; #location=devlocation; #buttonDisplay= objname ;template=1; deviceinstance=10006}}
{{#name=devdescription; #location=devlocation; #buttonDisplay= objname; template=1; deviceinstance=10007}}
{{#name=devdescription; #location=devlocation; #buttonDisplay= objname; template=1; deviceinstance=10008}}
{{#name=devdescription; #location=devlocation; #buttonDisplay= objname; template=1; deviceinstance=10009}}
{{#name=devdescription; #location=devlocation; #buttonDisplay= objname; template=1; deviceinstance=10010}}
{{#name=devdescription; #location=devlocation; #buttonDisplay= objname; template=1; deviceinstance=10011}}
{{#name=devdescription; #location=devlocation; #buttonDisplay= objname; template=1; deviceinstance=10012}}
{{#name=devdescription; #location=devlocation; #buttonDisplay= objname; template=1; deviceinstance=10013}}
{{#name=devdescription; #location=devlocation; #buttonDisplay= objname; template=1; deviceinstance=10014}}
{{#name=devdescription; #location=devlocation; #buttonDisplay= objname; template=1; deviceinstance=10015}}
{{#name=devdescription; #location=devlocation; #buttonDisplay= objname; template=1; deviceinstance=10016}}
{{#name=devdescription; #location=devlocation; #buttonDisplay= objname; template=1; deviceinstance=10017}}
{{#name=devdescription; #location=devlocation; #buttonDisplay= objname; template=1; deviceinstance=10018}}
```

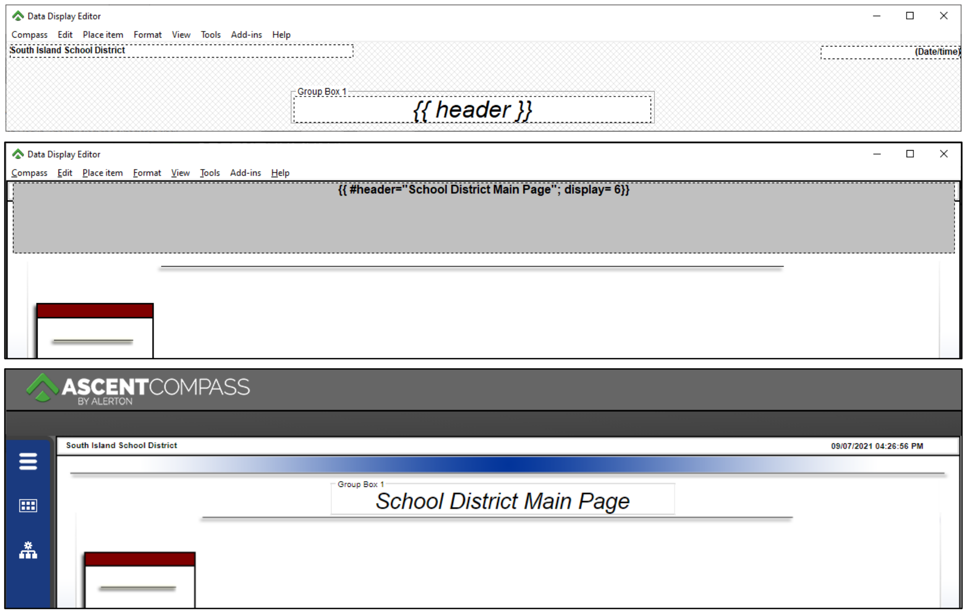
Figure 6 - Template Driven Quick View



{{display= 80000000 }}
{{descriptor}} - And Additional text
Area Served By
AHU-3
RTU-1
RTU-2
Main Gym
Display Info
ASCENTCOMPASS
SWHS First Floor Plan East - And Additional text

Figure 7 - Template Driven Floor Plan

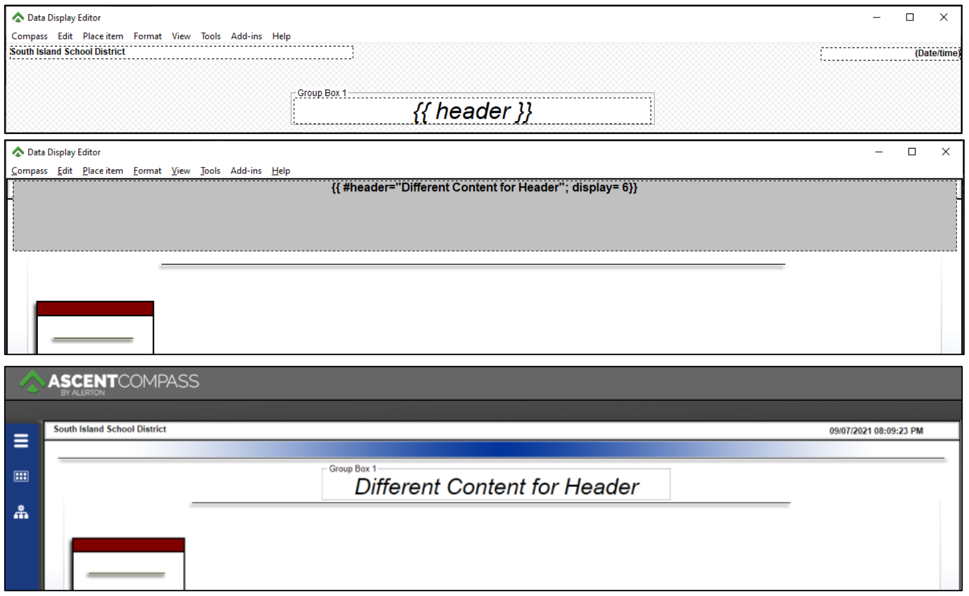
The above graphics are using **Nested Templates** (Px Compiler Feature) to build graphics. A major benefit of doing this is that copy/pasting elements from one graphic to many others is no longer needed. Anything that is often duplicated across multiple graphics is a good candidate as a nested template. Headers and navigation bars are good examples because they are often used on many displays. Using those things as templates means that they are created only once and if a change needs to be made on a template, then simply editing the template will update all other graphics that use that template.


Data Display Editor
Compass Edit Place item Format View Tools Add-ins Help
South Island School District
{Date/time}
Group Box 1
{{ header }}
Data Display Editor
Compass Edit Place item Format View Tools Add-ins Help
ASCENTCOMPASS
BY ALERTON
South Island School District
09/07/2021 04:26:56 PM
Group Box 1
School District Main Page

```
{{ #header="School District Main Page"; display= 6}}
```

Figure 8 - Variable declaration and assignment.
Top = nested template, Middle = destination display, Bottom = Compiled Display.

Other features when combined with templates can make it a powerful feature for creating quick views or floor plans. **Variable injection** is a feature that allows a user to define custom variables that can be used on a nested display, therefore enabling a nested display to be reused while containing unique content for each instance. Figures 8 and 9 show how to use variable injection to reuse nested templates. For instance, the top image in both figures is one nested template that uses a variable "header", but by assigning a different value to the "header" variable (middle images) the output display will be different (bottom images).


Data Display Editor
Compass Edit Place item Format View Tools Add-ins Help
South Island School District
{Date/time}
Group Box 1
{{ header }}
Data Display Editor
Compass Edit Place item Format View Tools Add-ins Help
ASCENTCOMPASS
BY ALERTON
South Island School District
09/07/2021 08:09:23 PM
Group Box 1
Different Content for Header

```
{{ #header="Different Content for Header"; display= 6}}
```

Figure 9 - Variable declaration and assignment.
Top = nested template, Middle = destination display, Bottom = Compiled Display.

There are two types of values that users can assign to variables: They are static strings, or dynamic strings. **Static strings** are text values enclosed in quotes that a user types out in the Compass graphics editor. Static strings were used in both Figures 8 and 9. **Dynamic strings** are not typed out by a user, rather their value is retrieved from the Bactalk database for a specific device property. This value is then assigned to the variable which becomes visible on the output graphic. For more details on how to use this feature, see Dynamic Variable Assignment in the Syntax and Commands section.

Figure 10 is an expression contained in a plain text object. This expression was used in Figure 6 to build a quick view. This expression tells Px Compiler to do a few things: First, a few variables are assigned. They are **"name"**, **"location"**, and **"buttonDisplay"**. These are dynamic string values, so they automatically assigned by Px Compiler from the Device Manager database. Next, **"template=1"** tells Px Compiler to place all the objects from device template 1 (00000001.dvtx) in the current display and position everything relative to the top-left corner of the plain text box. Last, **"deviceInstance=10005"** sets the device instance to use for this expression. Display objects on the nested template will be set to this device instance (if "use current device" is checked), and dynamic string values will use properties of this device instance.

#buttonDisplay is a special variable that is used to set pushbutton objects on the template to a specific number. This number can be a static value or dynamic value.

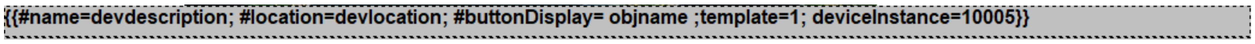

{{#name=devdescription; #location=devlocation; #buttonDisplay= objname ;template=1; deviceInstance=10005}}

Figure 10 - Expression & Syntax

After saving the display in Compass, Px Compiler will automatically compile the output display. Figure 11 shows what the output of the expression will look like after compilation.

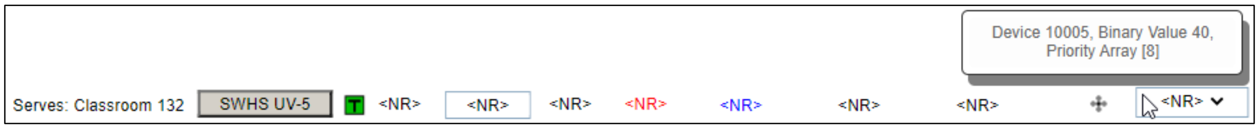


Figure 11 - Compiled Output from Figure 7

Placing the device instance last in the expression above was intentional. The **"Repeat item"** feature of Compass can be used to add more expressions on the page. Placing the device instance last allows the repeat operation to increment the device instance automatically. The same process can be used when creating floor plans.

The size and color of the plain-text box has no effect on the location of elements from a nested template. However, it is helpful to size the box appropriately so that other elements are not placed where there may be overlap with the nested template.

Syntax and Commands

This section will cover how to write expressions that Px Compiler can recognize. All commands and their syntax will also be explained here.

Syntax

All Expressions must be enclosed in double curly brackets **{{ … }}**. Most expressions are written inside of plain text box objects within Compass while some expressions are placed in the text field of a pushbutton object within Compass.

Semicolons **;** are used to combine multiple commands in one expression. Failure to use semicolons in this scenario will cause the compilation to fail.

Commands

Nested Display

Nested Displays place a source display onto a destination graphic. The destination graphic can be either a Compass display or device template.

The template is added to the destination graphic at the upper-left coordinate of the "plain text" box that contains the expression. The location of all elements in the template are relative to this point as well. Multiple templates can be used on a single destination graphic.

Only use this command once per expression, aka once per plain text box.

{{ display= 4 }}

This expression will place the contents of display *00000004.dspx* in the destination graphic. Leading zeroes are not required.

Nested (Device) Template

Nested Templates place a source device template onto a destination graphic. The destination graphic can be either a Compass display or device template.

The template is added to the destination graphic at the upper-left coordinate of the "plain text" box that contains the expression. The location of all elements in the template are relative to this point as well. Multiple templates can be used on a single graphic.

Only use this command once per expression, aka once per plain text box.

Typically, this command will need to be used with the Device Instance Assignment command for the desired effects.

{{ template= 4 }}

This expression will place the contents of display *00000004.dvtx* in the destination graphic. Leading zeroes are not required.

Device Instance Assignment (used with Nested Templates)

When using nested templates, usually it is necessary to also supply a device instance to the device template. All displays object in the Compass template that are set to " use current device" will use the supplied device instance. This command can only be used with nested device templates.

Only use this command once per expression, aka once per plain text box.

If placed last in the expression, the Compass repeat item feature can be used to auto-increment the number.

Separate all commands with a semicolon

{{ template= 4; **deviceInstance= 10005** }}

This expression will place the contents of display *00000004.dvtx* in the destination graphic and set the appropriate objects to target device instance 10005.

Variable Injection

Variables allow for adding custom text to nested graphics. Custom variables are defined in an expression within the destination graphic and are consumed on a nested graphic.

All variables must be preceded by a pound symbol, **#,** when declared. DO NOT use the pound symbol on the nested graphic.

Variables names must only consist of the characters: a-z, A-Z.

Multiple variables may be defined in an expression, but each must be separated by a semicolon.

Plain text values can be assigned to variables by using quotes. Special keywords do not require quotes. See Dynamic Variable Assignment for more details.

{{ **#NewVariable="Some Text";** template= 4; deviceInstance= 10005 }}

The variable **NewVariable** is defined and assigned the value of **Some Text**. All instances of the **NewVariable** in the nested template will be replaced with the value **Some Text**.

To consume variables on the template, create an expression containing that variable inside of the text field of objects. The pound symbol MUST be omitted.

Variables may also be placed in the text field of a Compass pushbutton object OR a Compass plain text object.

Variable = {{ NewVariable }}

Continuing from the previous example; When the above is entered into a plain text box, the output graphic will display **Variable = Some Text**. The text that is outside of the double curly brackets is not affected.

Dynamic Variable Assignment

Variables may also be assigned the value of any properties that are present in the **tblDevices** table of the project Database (See BACtalk.mdb or SQLEXPRESS for all available properties). This is accomplished by entering the name of the desired field, exactly.

Do not include quotes when using these special properties.

A device instance must be assigned in the same expression for this to work.

{{ **#NewVariable=objname;** template= 4; deviceInstance= 10005 }}

The value of **objname** for device instance 10005 is assigned to variable **NewVariable**. In this example. The value of objname is **UH-5**.

Some of the common properties from the database that may be used are: **DDC_app, devdescription, objname, devlocation**.

The values will only be assigned when the graphic is saved in Compass. If the value of the property is changed in the database, the graphic must be saved again to get the updated value.

Dynamic Pushbutton Assignment (used with Nested Device Templates)

When using nested templates with pushbutton objects, specifying the target display of the button is possible. This is helpful when using a single nested template for multiple devices that have unique equipment displays.

#buttonDisplay is a reserved variable that can be assigned a number OR use a Dynamic Variable Assignment.

{{ **#buttonDisplay=1000;** template= 4; deviceInstance= 10005 }}

This will set the applicable pushbutton objects to target display number 00001000.

{{ **#buttonDisplay=objname;** template= 4; deviceInstance= 10005 }}

This will set the applicable pushbutton objects to target display number 00002000. In this scenario, the number value is assigned the value of the objname property from the database for device instance 10005.

{{ buttonDisplay }}

Placing the above command in the text field of a pushbutton object on the nested template will direct Px Compiler to reassign the buttons display number. Text outside of the double curly bracket will not be affected, and the expression text will not be visible on the output graphic.

{{ buttonDisplay, NewVariable }}

This command may be combined with another variable, but they must be separated by a semicolon. The expression text will be replaced with the value of the variable **NewVariable**.

Display Descriptor Reference

There are two keywords that can be added to a display or template which will automatically add the Compass display descriptor to the output graphic.

These commands can only be placed in a plain text object.

The display descriptor must be set from the Compass program window. No text will appear if the descriptor is blank.

{{ descriptor }}

This will place the value of the descriptor in the plain text object. Any text outside of the double curly brackets will not be affected. The descriptor of the graphic that this command resides on will be used.

{{ destDescriptor }}

This will place the value of the descriptor in the plain text object. Any text outside of the double curly brackets will not be affected. The descriptor of the destination graphic will be used. I.e., This command, when used on a nested template, will be replaced with the descriptor of the destination graphic (NOT the descriptor of the nested template).

Changelog

v1.4.20

Added

- Application version check and auto-updating.

v1.4.19

Added

- Loading spinner for export operation.
- Styling to Device Table header tooltips

Changed

- Added restrictions to evaluation mode.
- Windows icon to improve legibility.

Fixed

- BACnet interface settings would not save when navigating around application screens.
- Device Table would load from database twice on initialization.
- Improved variable injection support.

v1.4.18

Added

- New button which re-compiles all displays in current directory.
- New feature: “Track DB Changes” automatically re-compiles specific displays when the database changes.

v1.4.17

Added

- New Feature (still in Beta). BACnet interface added which allows the user to read/write BACnet values to devices from within the Device Table. The following properties are supported: objname, devdescription.
- Added “destDescriptor” command.

Changed

- Moved the Device Table toggle buttons to a side navigation bar.

vX.X.X

Added
Changed
Deprecated
Removed
Fixed
Security