Technical Document

Niagara Templates Guide

Oct 4, 2021

niagara4

Niagara Templates Guide

Tridium, Inc.
3951 Westerre Parkway, Suite 350
Richmond, Virginia 23233
U.S.A.

Confidentiality

Trademark notice

Copyright and patent notice

Contents

About this Guide **5**
Document change log 5
Related documents 6

Chapter 1 Overview **7**
Latest enhancements 7
Enhancements for earlier releases 7

Chapter 2 Template creation **9**
Making a template 9
Modifying the component tree 11
Configuring exposed properties 12
Adding relations 12
Managing I/O links 13
Managing graphic views 15
Adding sub-templates 17
Updating the parent when a sub-template changes 17
Removing subtemplates 19
Editing a template 19
Creating a station template 19
Making a template module 21

Chapter 3 Template deployment **23**
Deploying a template via drag-and-drop 23
Deploying a template via the Device Manager 26
The provisioning of deployed templates 28
Upgrading a deployed template 28
Downgrading a deployed template 29
Redeploying a deployed template 29
Detaching a deployed template 30

Chapter 4 Application Template **31**
Application Template vs. Station Template 31
Application Template vs. Component (Device) Template 31
Existing file and component structures for templates 32
Creating an application template 33
Installing an application template 34
Upgrading an application template 37
Optional components in application templates 39
Establishing optional components 39
Installing optional components 41

Chapter 5 Template bulk deployment **43**
Exporting the template spreadsheet 44
Editing a spreadsheet 46
About the exported spreadsheet 47
Template instance data 48

Template inputs, outputs, relations, configs, and tags....48
Deploying bulk templates....50
Exporting the application template spreadsheet....52
Editing the application template spreadsheet....53
Installing bulk application templates....54
Updating template configuration....55
Optional components in application templates....57
Optional components when installing bulk application templates....57

Chapter 6 Components and plugins....59
TemplateService component....59
TemplateConfig component....60
Template sidebar....60
Template Manager view....61
Additional manager functionality....63
Provisioning commands for deployed templates....65
About the provisioning Deploy process....65
About the provisioning Detach process....67
Template view....67
Template Info tab....67
Component tab....69
Configuration tab....69
Relations tab....70
Template I/O tab....71
Graphics tab....71
Subtemplates tab....72
Template Options....73
Example template.manifest.xml file....74
Template troubleshooting....75

Glossary....77

Index....79

About this Guide

This topic contains important information about the purpose, content, context, and intended audience for this document.

Product Documentation

This document is part of the Niagara technical documentation library. Released versions of Niagara software include a complete collection of technical information that is provided in both online help and PDF format. The information in this document is written primarily for Systems Integrators. To make the most of the information in this book, readers should have some training or previous experience with Niagara software, as well as experience working with JACE network controllers.

Document Content

This document describes how to create and use templates to improve productivity and consistency when working with station and enterprise systems. Sections in this guide include chapters on common template tasks, template concepts, and reference information. Also included are images and descriptions of the primary software user interface windows involved when working with templates.

Document change log

Updates (changes and additions) to this document are listed below.

October 4, 2021

Added minor edits.

May 6, 2021

Minor correction to the Note added in previous edits.

April 16, 2020

Added a note to several topics explaining the need to copy relevant tag dictionaries to the User Home tagDictionary folder prior to making, editing, or deploying a template.

Minor edits through out plus added information on enhancements for Niagara 4.9. Also in the in the Application Templates chapter, added procedures for upgrading and updating application templates.

December 11, 2019

Many edits throughout related to functional changes (in Niagara 4.9 and later) for optional components in application templates.

June 16, 2019

Added the "Template Options" topic to the chapter, "Component and Plugins".

April 10, 2019

In the chapter, "Template bulk deployment", edited several topics updating those for the latest functional changes.

March 18, 2019

Revisions related to the Niagara 4.8 release include adding the following topics to the "Template bulk deployment" chapter:

- "Exporting the application template spreadsheet"
- "Editing the application template spreadsheet"
- "Deploying bulk application templates"

October 2, 2018

In the "Overview chapter", added "Enhancements in Niagara 4.7". Edited template creation topics, "Managing I/O Links" and "Deploying a template via drag-and-drop", to reflect changes in functionality. Also there are updates to a few topics in the "Application Template" chapter.

August 14, 2018

Document updated for Niagara 4.7 that includes the Application template feature.

May 29, 2018

Document updated for Niagara 4.6 including the template bulk deployment feature. Minor correction in the topic, "Detaching a deployed template."

April 4, 2017

Edited the procedure, "Creating a station template," to include subtemplates.

March 22, 2017

Edited the topic," Example template.manifest.xml file," to add a description of the template id number.

March 9, 2017

This document has been updated to cover the template enhancements in Niagara 4.3 which include support for versioning templates, upgrading deployed templates as well as deploying and provisioning subtemplates in a parent template's config bog. Additionally, there are many minor changes throughout the document, and several added topics on the new functionality.

November 18, 2015

Minor updates

August 13, 2015

Initial release document

Related documents

Following are documents related to this guide.

- Niagara Hierarchies Guide
- Niagara Provisioning Guide
- Niagara Relations Guide
- Niagara Tagging Guide

Chapter 1 Overview

Topics covered in this chapter

♦ Latest enhancements
♦ Enhancements for earlier releases

Niagara Template is a deployable package of Niagara objects. The purpose of a template is to allow a set of configured objects to be encapsulated and deployed as a unit. The goal is to eliminate repetitious configuration steps when making the multiple installations with similar functionality.

Templates contain a component tree with a single base component and associated support objects. A template is contained in a file that has a `.ntpl` file extension. Template files may be grouped into modules for ease of distribution. Also, one or more instances of a template may be deployed to a station.

The deployment process extends the normal installation processes used for a single component. At deployment you only need to make modifications unique to that installed instance.

A template provides mechanisms to designate certain properties in the template component tree as special configuration properties or linkable input/output properties. Configuration properties may be set to customize different deployed instances of the template. Input/output properties must be linked after deployment.

Template bulk deployment applies one or more templates to a single station. Provisioning a template bulk deployment job applies one or more templates to multiple stations.

Latest enhancements

Summary of latest enhancements to templates.

Niagara 4.9 enhancements

Niagara 4.9 enhancements include the following:

- Optional components in application templates for manual install, install via bulk deploy, and install via provisioning;
- Upgrading application templates manually and via provisioning;
- Updating application template or component template configuration from spreadsheet manually and via provisioning.

Niagara 4.8 enhancements

In Niagara 4.8 and later, there is added support for installing application templates using bulk deployment. The workflow is very similar to component and device templates, with a few key differences that are outlined below.

For details, see these topics in the "Template Bulk Deployment" chapter:

- "Exporting the application template spreadsheet"
- "Editing the application template spreadsheet"
- "Deploying bulk application templates"

Enhancements for earlier releases

Application templates enhancements in Niagara 4.7

In Niagara 4.7 and later, there is added support for application templates.

Application templates are a mechanism to deploy an entire station application to a running station. They are the primary means of sharing solutions that are built for Edge devices. In concept, application templates can be used wherever a whole station needs to be replicated and installed to running stations.

For details, see the "Application Templates" chapter of this guide.

Bulk deployment enhancements in Niagara 4.6

In Niagara 4.6 and later, there is added support for templates deployed in bulk, that is multiple templates can be deployed to a station in a single operation.

The **Template Manager** view has a **Bulk Deploy from Excel** button to deploy bulk templates from an Excel spreadsheet.

For details, see the chapter on "Template bulk deployment".

Template enhancements in Niagara 4.3

In Niagara 4.3 and later, there are a number of template enhancements. Specifically, templates now support subtemplates, offline station template deployment, as well as deployed template provisioning.

Subtemplate support

It is possible to configure a template that contains other templates. A template contains a list of any subtemplates and the **Template View** supports the management of any included subtemplates.

- The **Make Template** menu option has a **Subtemplates** tab that shows any contained subtemplates. This is the same **Template Manager** view used by the **TemplateService** that supports the provisioning of deployed templates.
- The **Template View** has a **Subtemplates** tab that shows any contained subtemplates, This is the same **Template Manager** view that is used by the Template Service that supports the provisioning of deployed templates.
- The **Template** sidebar has an added **Find Usage** option on the right-click menu. This option provides a list of parent templates that contain this template.

Offline station template deployment support

In Niagara 4.3 and later, it is possible to execute the NEQL searches required to support this. The deployment and provisioning work flow is the same as the online station work flow.

Deployed template provisioning support

In Niagara 4.3 and later, the **Template Manager** view is changed to support provisioning deployed templates. In prior releases the **Template Manager** view was the default view of the **TemplateService**. It provides the visualization of the deployed templates and indicates any unresolved inputs, outputs, or relations. It also provides a means for the user to initiate to resolve any unresolved inputs, outputs, or relations. In Niagara 4.3, the **Template Manager** view is still the default view of the **TemplateService**, but it is also used by the **Template View** to manage the subtemplates deployed in a template.

Chapter 2 Template creation

Topics covered in this chapter

♦ Making a template
♦ Editing a template
♦ Creating a station template
♦ Making a template module

Configuring the device properties for a specific group of identical or similar devices is the ideal situation in which to use a template. In this situation, you would make a device template. Do this just after you finish configuring properties for the first device.

As an example, assume you are configuring four air handler units (AHUs), one for each floor of a four-story building and the temperature limits vary from floor to floor. Using an AHU component that you have already tagged and configured for the first floor, you can make an AHU template.

The tabs in the **Template** view allow you to provision the template as follows:

- On the **Components** tab — modify the template's components and links as needed.
- On the **Configuration** tab — select and expose the temperature limits.
- On the **Relations** tab — define any desired relations between the root component of this template and other components in the station in which the template is deployed.
- On the **Template I/O** tab — set up the required links for the template inputs and outputs.
- On the **Graphics** tab — create or modify the graphic views associated with the components in the template component tree.
- On the **Subtemplates** tab — manage and provision related sub-templates contained by the parent template.
- When finished, save the template — the system saves a new template file (`.ntpl`) with the default name or the name you entered on the **Template Info** tab, and stores the file in your Workbench User Home`/templates` directory. Once saved, the template is available from the **Template** side bar.

Using the **Device Manager**, you can then apply the AHU template to the discovered AHU devices for the remaining floors. During actual template deployment you are prompted to set the temperature limits and resolve the inputs, outputs, and relations for each device.

NOTE: To avoid future problems, make sure that you tag and configure devices correctly before making a template and using it to configure other devices. Additionally, you must copy all relevant tag dictionaries (with the exception of the Niagara tag dictionaries) from the tag dictionaries Service to the tag dictionaries folder in your User Home file space before the template is first created (or edited), and during template deployment to be successful. Also, the system does not automatically apply subsequent template changes to the configured devices.

Making a template

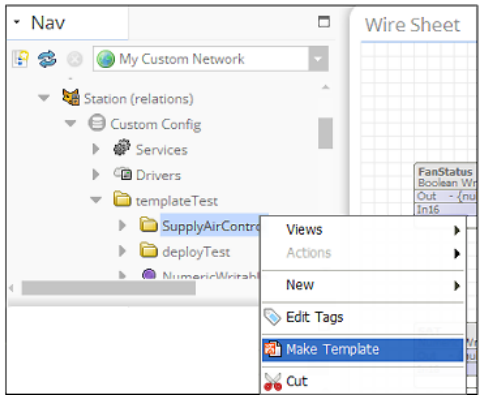
Making a template is the first of several procedures in the template creation process. This section documents the steps to be done on each of the six configuration tabs in the **Template** view.

Prerequisites:

- The device driver is located under the **Config→Drivers** node in the Nav tree.
- The device is located under the driver network.
- You have assembled a collection of tagged control logic including self-contained graphic view(s) with a single component as the root.

Step 1 In the Nav tree, expand the **Config→Drivers** container.

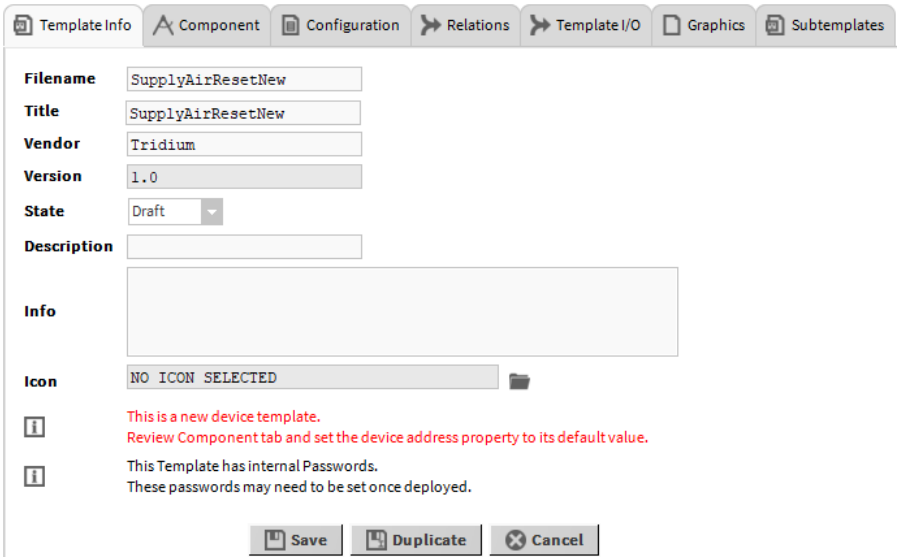
Step 2 Right-click on the desired root component (shown here, the root component is a folder), and click **Make Template**.



The newly created template of the selected root component appears in the **Template View**, with the **Template Info** configuration tab selected. By default, the template name is the same as the root component.

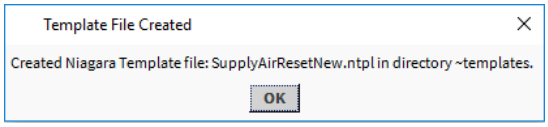
Step 3 Edit the following information on the tab:

- **`Filename`** - edit the default filename to enter a unique name.
- **`Title`** - enter the preferred name for the template.
- **`Description`** - enter a short description of the template.



Step 4 To create the template, click **Save**.

The **Template File Created** window opens and displays the path of the created template file.



Step 5 To close the Template file Created window, click **OK**.

Any of the following notices may display on the **Template Info** tab:

- If the template component tree contains links whose source or target components are not contained within the same component tree, a notice alerts you to review the automatically added I/O links and associated **`bindHints`** in the **Template I/O** tab.
- If the template root component is a Device component, a notice alerts you to review the **Component** tab, and to set the device address property to its default value.
- If the template component tree contains a **`Password`** property, a notice alerts you that during deployment the user may need to set the password.

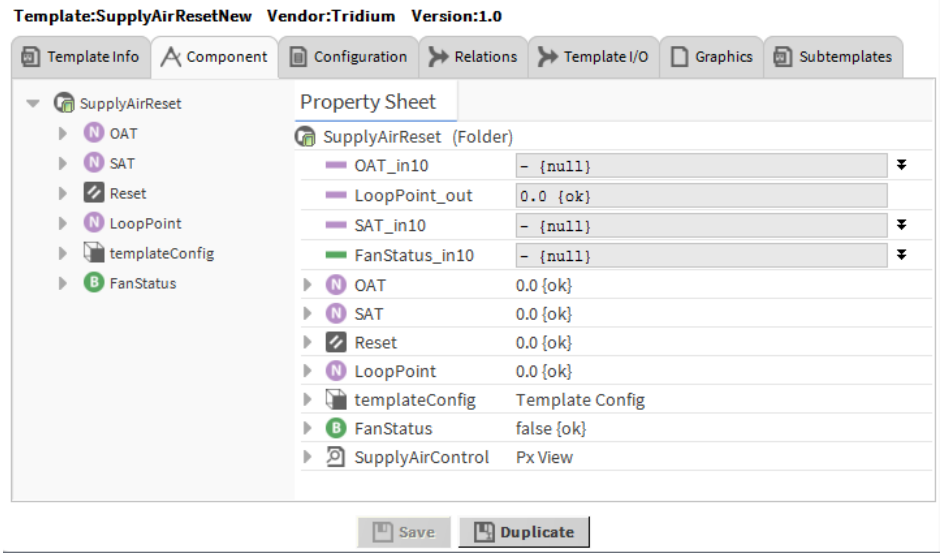
Modifying the component tree

Modifications to the component tree change the template's control logic by adding and removing components, as well as adding and removing links.

Step 1 In the **Template View**, click the **Component** tab.

Step 2 In the left pane, right-click the template root component to select a view (`Property Sheet`, `Wire Sheet` or `Slot Sheet`) of the component.

The selected view displays in the right pane.



Step 3 To add a component, drag it from the Nav tree or a palette to the root component of the template in either the left or right pane.

Step 4 To delete a component from the component tree, right-click it in the tree and click `Delete`.

Step 5 To mark a component in preparation to create a link between two components, right-click the source or target component and click `Link mark`.

This system marks the component.

Step 6 To create the link, right-click the other component and select either `Link From` or `Link To`.

Where the `link-marked component` is either the source or target of the link.

Step 7 To remove a link, on the **Wire Sheet** view of the root component of the template, right-click the link you wish to remove and click `Delete Links`.

Step 8 Before you go to the next tab, click **Save**.

Configuring exposed properties

The **Configuration** tab in the **Template View** is to select and expose specific component properties contained within the template control logic. During deployment a popup prompts the user to edit the values for these properties.

Step 1 In the **Template View**, click the **Configuration** tab.

Step 2 In the left pane, expand components until the desired property is visible.

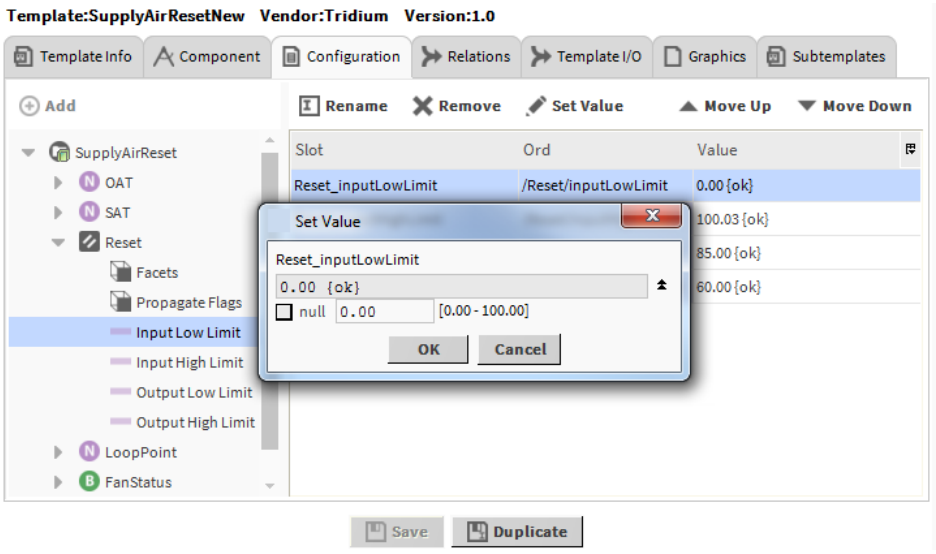
Step 3 To expose an individual property, double-click it.

The system adds the property to the table in the right pane.

Step 4 Optionally, to rename a property, double-click on the **`Slot`** name cell.

Step 5 To edit the default value for an exposed property, double-click on the a **`Value`** cell or select the row and click on **Set Value**).

The **Set Value** window opens.



Step 6 Enter the default value and click **OK**.

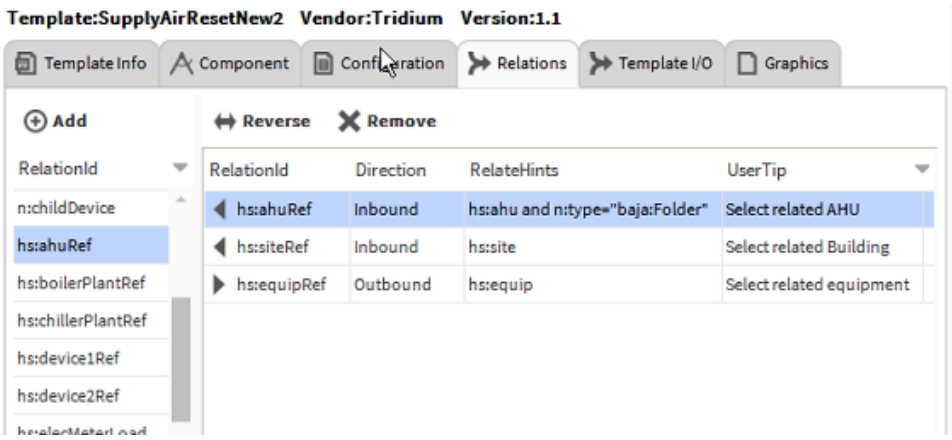
Step 7 For each exposed property, repeat these steps, then click **Save**.

NOTE: The value entered into the template for each exposed property becomes the default value upon deployment. The deployment job prompts the user to edit this value.

Adding relations

The **Relations** tab in the **Template View** is to define the desired relations between the root component of the template and other components in the station in which the template is deployed.

Step 1 Click the **Relations** tab.



Step 2 Add a relation using either of these methods:

- To add from the installed tag dictionaries, select a **RelationId** in the left pane and click Add (or double-click the **RelationId**).
- To add an ad hoc relation (one not found in a tag dictionary), with nothing selected click Add and enter the desired **RelationId** in the popup window.

The added relation appears in the right pane.

Typically, the default **Relation Direction** is sufficient. However, you may need to change a relation direction to get the desired query results, that is, hierarchy results.

Step 3 To change the relation direction, in the right pane, select the row containing the added relation and click Reverse .

You can add a given **RelationId** only twice. The first time you add a relation, the default relation direction is inbound. If you add the same **RelationId** a second time, by default it has the opposite relation direction.

Step 4 To set up a NEQL query, in the right pane, double-click on the **RelateHints** cell in the **Set Value** window, enter a NEQL search predicate value, and click **OK**.

This step tags the relation with a string value, which the system uses as a NEQL search predicate during the deployment to search for and suggest potential related components.

For example, if you add the `hs:ahuRef` **RelationId**, you might enter this search predicate: `hs:ahu`.

If your entry is a valid NEQL search predicate, the system updates the table in the right pane with this value. If it is not valid, the system returns an error message in the table.

Step 5 To provide relation guidance for the person deploying the template, double-click on the **UserTip** cell of the row, and, in the **Set Value** window, enter a instruction text, and click **OK**.

For example, you might enter: `Select Related AHU`.

This updates the table in the right pane with the **UserTip**.

Managing I/O links

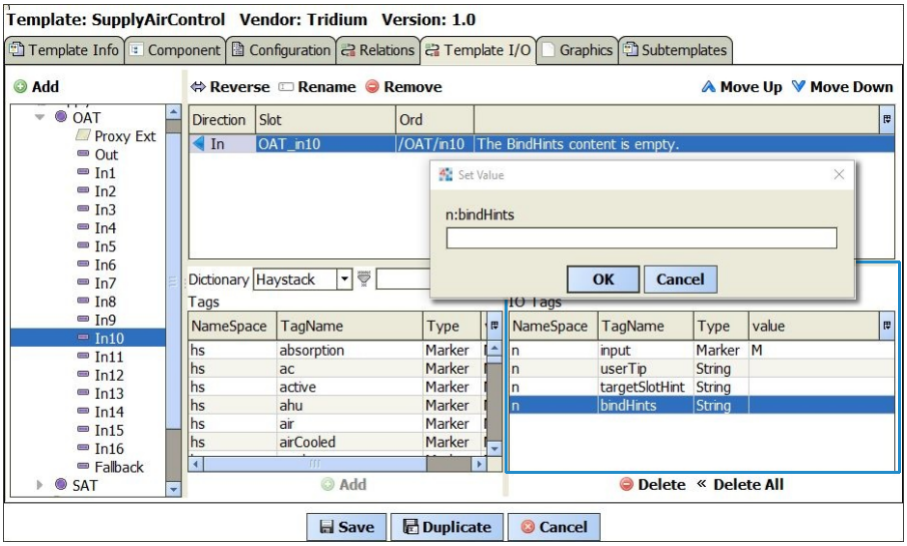
The **Template I/O** tab allows you to expose specific control inputs required and control outputs provided by the template. Also, (if licensed for **tags** and **tagdictionaries**) you can apply additional tags to the exposed I/Os. During deployment the template attempts to resolve these linkable I/O properties, and may prompt the user to choose among multiple sources.

This procedure describes adding a control input to expose it as a linkable point in the template.

Step 1 In the **Template View**, click the **Template I/O** tab.

Step 2 To add an input, in the left pane click to expand the control point and double-click on the input slot or select the input slot and click the **Add** button.

The system adds the input slot to the template's slot sheet shown in the upper right pane.



By default, the system automatically adds several tags (**`n:input`**, **`n:userTip`**, **`n:targetSlotHint`**, and **`n:bindHints`**) to the **I/O Tags** pane (lower right) along with any other tags previously applied to this input slot. These added tags are used during template deployment to assist the person deploying the template in linking the template inputs from other writable points in the station.

- **`n:bindHints`** is a string value that is used as a NEQL search predicate. In the example shown, the selected input is for the current outside air temperature and uses the **`HaystackTagDictionary`**.
- **`n:targetSlotHint`** is a string value that is used to suggest the target slot of any writable points that match the **`n:bindHints`** query. The value is an output. For example, if the targetSlotHInt tag value is: "out", the deploy tool would first suggest linking from source slot out if it was linkable., If it is not linkable, then it does not suggest any particular output, it lets the user choose. In any case, the person deploying the template can override the suggested target slot.
- **`n:userTip`** is an additional string value that appears in a popup window when the template is deployed. This provides guidance to the person deploying the template in making a selection.

Step 3 In the upper right pane, select the added input slot to edit any of the added tags.

Step 4 To edit the **`n:bindHints`** tag, in the **I/O Tags** pane (lower right), select the **`n:bindHints`** row and double-click on the **`Value`** cell.

The **Set Value** window opens.

Step 5 Enter a NEQL search predicate for the input slot and click **OK**.

TIP: Use the tag dictionaries in the lower left pane to identify tags to include.

In the example image above, the NEQL search predicate entered as the value for **`n:bindHints`** is: `hs:outside`, and `hs:air` and `hs:temp` and `hs:sensor`.

The system validates the **`n:bindHints`** value when you enter it and indicates any errors in the **I/O Tags** pane (lower right).

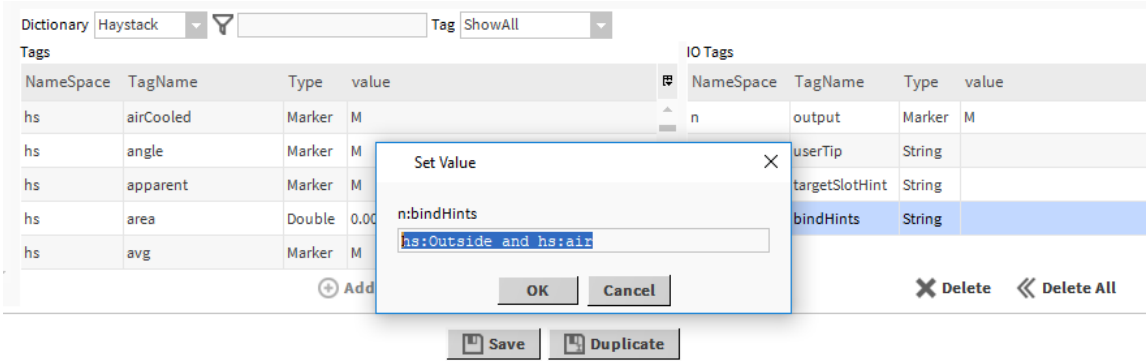
Step 6 To create a template output, in the left pane click to expand the control point and double-click on the output slot to add it (or select it and click **Add**).

NOTE:

This is basically the same procedure as creating a template input, except that you typically select the "out" property of a control point.

The system adds the output slot to the template's slot sheet, and displays it in the upper right pane.

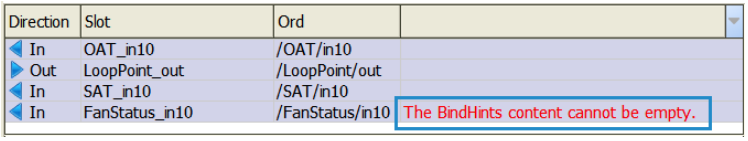
By default, the system automatically adds the **n:output**, **n:bindHints**, **n:targetSlotHint**, and **n:userTip** tags and displays them in the **I/O Tags** pane (lower right). The deployment job uses these tags to assist the person deploying the template in linking the template outputs to other writable points in the station.



- **n:bindHints** is a string value that the system uses as a NEQL search predicate.
- **n:targetSlotHint** is a string value that is used to suggest the target slot of any writable points that match the **n:bindHints** query. The value is an output indicated by a comma separated list of inputs. The order of the list is the order that the deploy tool uses to suggest the target slot. For example, if the **n:targetSlotHint** tag value is: `in10, in13`, the deployment job first suggests linking to the target slot `in10` if it is linkable. If it is not linkable, the deployment job suggests `in13`. If neither is linkable, the deployment job does not suggest an input slot. Instead. it lets the person deploying the template choose. In any case, the user can override the suggested target slot(s).
- **n:userTip** is an additional string value that appears in a popup window when the template is deployed. This provides guidance to the person deploying the template in making a selection.

Step 7 To edit the **n:bindHints**, **n:targetSlotHint**, and **n:userTip** tags in the **I/O Tags** pane (lower right), select the tag row and double-click on the **Value** cell, edit the value in the **Set Value** window, and click **OK**.

The system updates the table (upper right pane) with the entered values. The value is validated upon entry and any errors are indicated in the I/O table.



Step 8 At this point you can expose additional inputs and outputs, as needed.

On completion, the template's linkable inputs and outputs are configured.

Managing graphic views

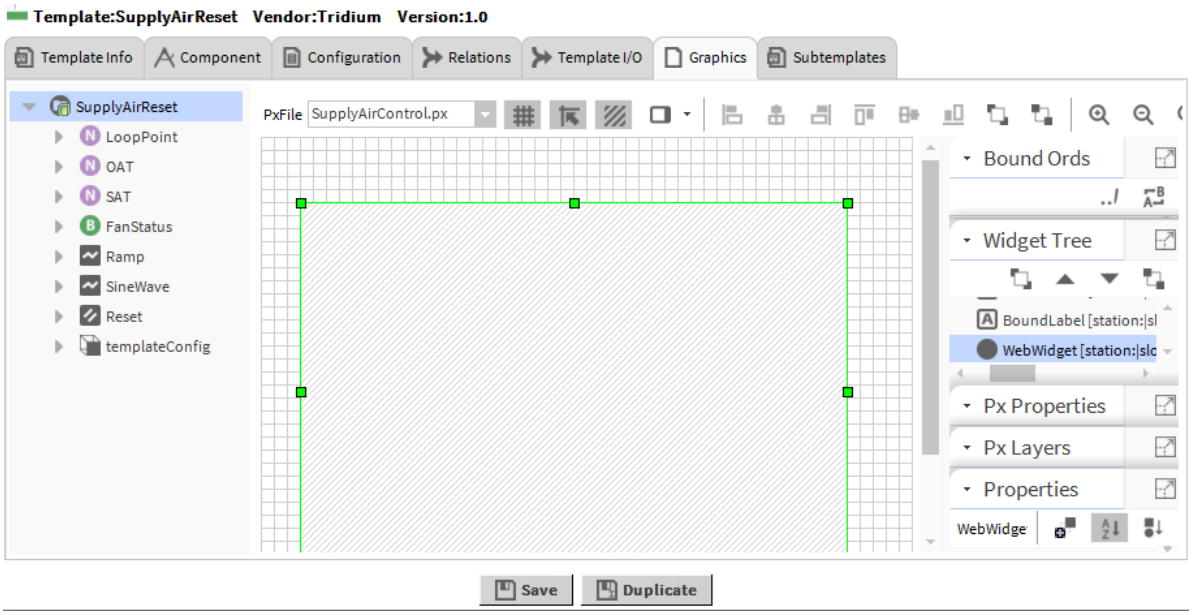
The **Graphics** tab in the **Template View** allows you to create or modify the graphic views contained in the template.

Prerequisites: The graphics to use in the template already exist. All value bindings in graphics are relative to the root component of the template.

NOTE: A best practice for creating and editing graphics for templates is to fully engineer them in a station before making the template. This is especially true if image files are used as they will likely not be visible in the template graphic editor. Images from `.jar` files are visible in the template editor. Use the template graphic editor to make minor changes to graphics.

Step 1 In the **Template View**, click the **Graphics** tab.

The **Graphics** tab opens.



This tab is a wrapper view of the **PxEditor** view. The left pane is the Nav pane and the right pane is the **PxEditor** pane. Selecting a PxFile loads the file into the editor and selects the component(s) in the Nav pane that have a view using the selected PxFile.

The toolbar at the top of the **PxEditor** pane, shown here:

Step 2 In the right pane, click the **PxFile** drop down list and select one of the PxFiles to include in the template.

Step 3 Make minor editorial changes to the graphic, as needed.

You can drag items from the nav pane (left) to the edit pane (right).

NOTE: All value bindings must be relative bindings. The bindings are forced to be relative to the root component of the template.

Step 4 To complete the procedure, click **Save** or **Save As**.

This saves your changes to the PxFile and saves the configured template, creating a template `.ntpl` file.

The system saves a new template file with the default name or the name you entered on the **Template Info** tab, and stores the template in your Workbench User Home`/templates` directory. Once saved, the template is available from the **Template** side bar.

IMPORTANT: Be sure to test your template on another component of the same type and modify the template as needed before using it to configure other devices—especially to bulk-configure multiple Edge devices. Niagara does not automatically apply subsequent template changes to already-configured devices.

Adding sub-templates

In Niagara 4.3 and later, you may add one or more sub-templates to a template in the same manner as deploying a template in a station except that you are deploying it to the parent template bog file.

Prerequisites: You have at least two fully configured, saved templates. One is the parent template and other is the sub-template.

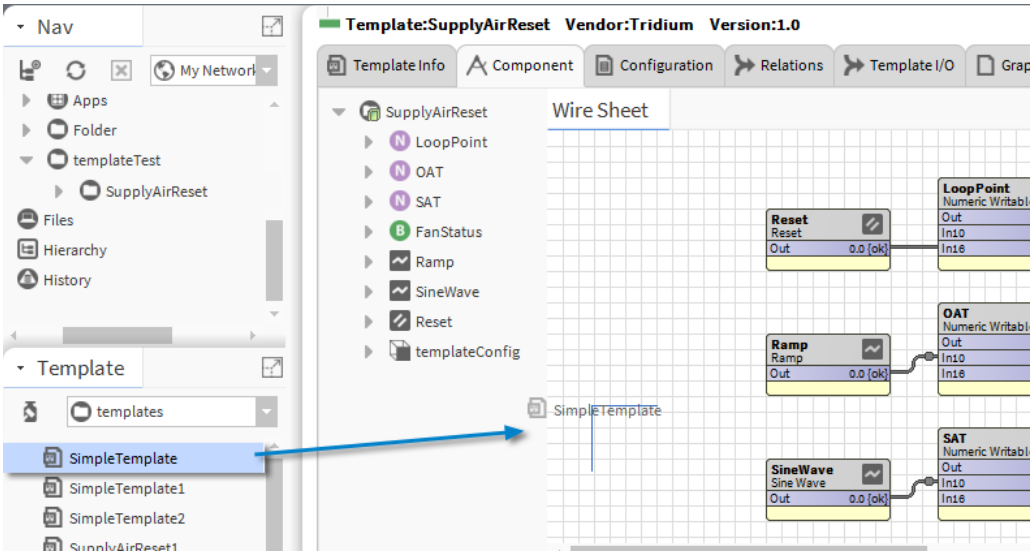
Step 1 Open the **Template** sidebar by clicking **Windows→Side Bars→Template**, and double-click the template that is the parent template.

This opens the **Template View**.

Step 2 Click the **Component** tab in the parent template.

The **Wire Sheet** view opens.

Step 3 In the **Template** sidebar, drag the sub-template to the **Wiresheet**.



At this point, the **Template** prompts you to resolve any inputs, outputs, and relations for the added sub-template.

Step 4 Respond to the prompts to resolve the I/Os and relations.

Step 5 To save the template relationship, click **Save**.

The added sub-template is visible in the **Subtemplates** tab.

Updating the parent when a sub-template changes

When a template is deployed it always uses the latest version of any contained sub-templates found on the client Workbench, even if the parent template has not upgraded its sub-templates. A modified template or sub-template may include additional inputs, outputs, relations, or configuration properties. For that reason,

it is important to review any templates that contain a modified sub-template so that you can resolve any new I/Os within the parent template.

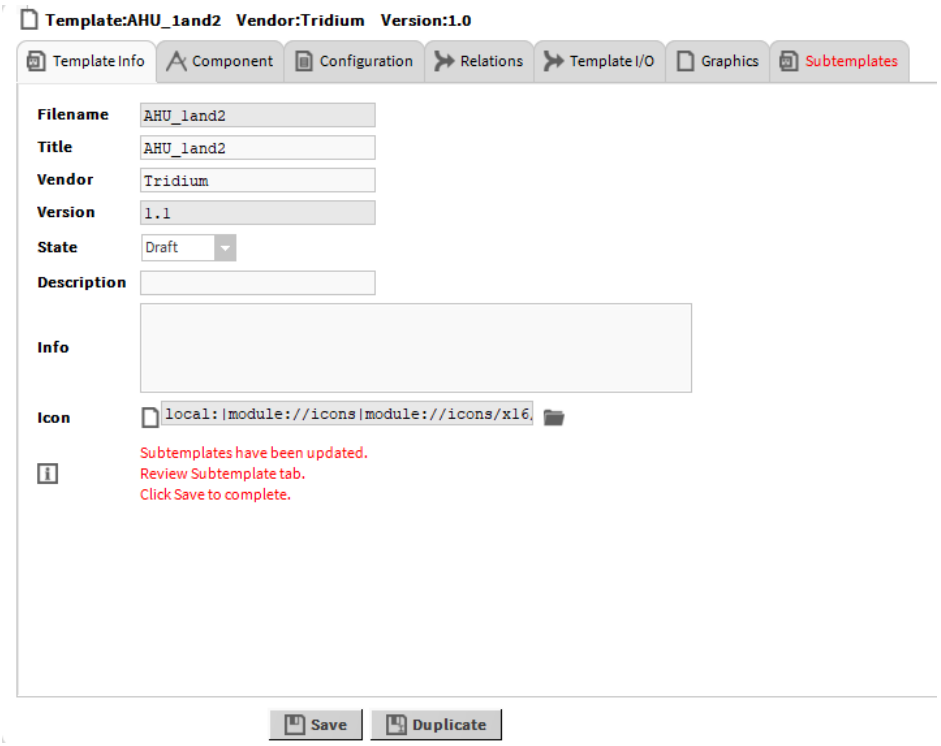
Step 1 Right-click a sub-template and click **Views→Find Usage**.

The system displays one or more parent templates that contain the sub-template.

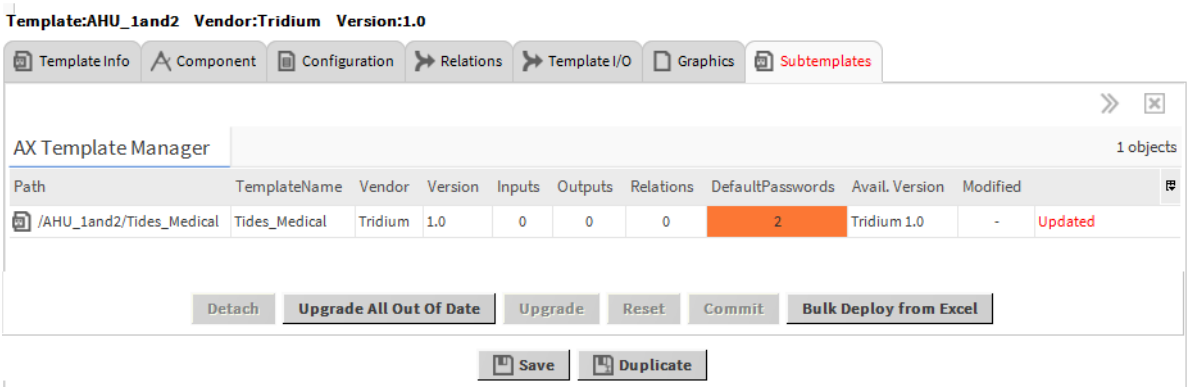
Step 2 Double-click the parent template in the **Templates** side bar.

The **Template View** opens.

In Niagara 4.3 and later, when the **Template View** loads a template, it checks for sub-template instances and determines if newer versions are available on the Workbench PC. If it determines that there are out-of-date sub-templates, it highlights the **Subtemplates** tab (with red text) and displays an advisory message on the **Template Info** tab.



Step 3 Click on the **Subtemplates** tab to see which templates have been changed (updated).



Any sub-templates that have been updated are flagged as `Updated` on the **Subtemplates** tab.

Step 4 To update the parent template click **Save**.

The sub-template status changes to `Up to Date`.

Removing subtemplates

You can remove a subtemplate simply by removing its root component from the Template component space.

Prerequisites: An existing, fully configured template that contains a subtemplate.

Step 1 In the **Template** sidebar, double-click on the parent template to open the **Template View**.

Step 2 On the **Component** tab, right-click on the subtemplate root component and click **Delete**.

The subtemplate is removed from the **Subtemplate** tab.

Editing a template

The **Template** side bar provides access to all template files located in the templates, `stationTemplates`, and modules directories.

Prerequisites: The template to edit exists.

NOTE: To avoid future problems, make sure that you tag and configure devices correctly before making a template and using it to configure other devices. Additionally, you must copy all relevant tag dictionaries (with the exception of the Niagara tag dictionaries) from the tag dictionaries Service to the tag dictionaries folder in your User Home file space before the template is first created (or edited), and during template deployment to be successful. Also, the system does not automatically apply subsequent template changes to the configured devices.
All templates have a file extension of `.ntpl`. Template and station template files are stored in the Workbench User Home `templates` and `stationTemplates` directories, respectively.

Template modules have a file extension of `.jar` and are stored in the Sys Home `modules` directory. A template module must be expanded in order to access the template file(s) within.

Step 1 To locate a template, open the **Template** side bar by clicking **Window→Side Bars→Template**.

Step 2 In the **Template** side bar, click the pull down menu and select either: the `Template` or `Module`.

NOTE: To see templates stored in a template module, select the `myModule` folder and expand the module. You cannot edit a template stored in a module. When you open it, `ReadOnly` appears in the top left corner of the **Template View**. To make changes you must first click **Save As** and save it as a new template in the `templates` folder.

Step 3 Double-click on the template.

The **Template View** opens that displays the template tabs with the **Template Info** tab selected.

Step 4 To modify components and properties, click the **Configuration** tab in the **Template View**.

Step 5 To save your changes to the template when finished, click **Save** or, click **Save As** to create a new variation of the template with a different filename (leaving the original template unchanged).

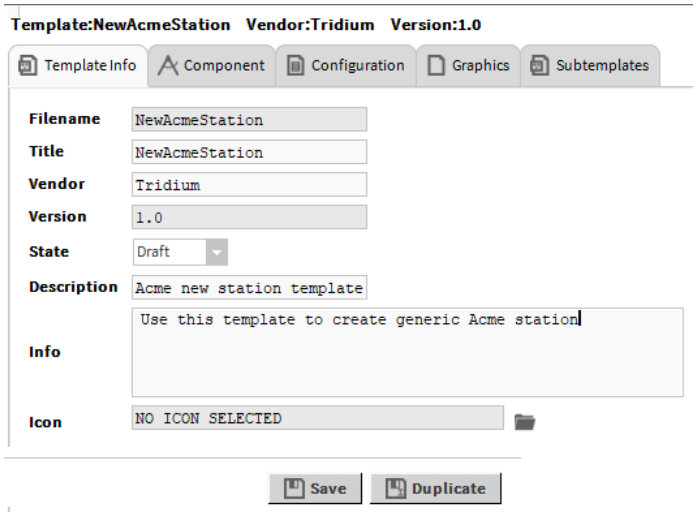
Creating a station template

With a station template that contains everything needed for the initial starting point of a new station, you can configure multiple stations in a single step. This procedure is one the Systems Integrator might perform on a fully configured basic station for purposes of reuse and standardization.

Prerequisites: An existing, fully-configured station suitable for use in a generic new station template exists.

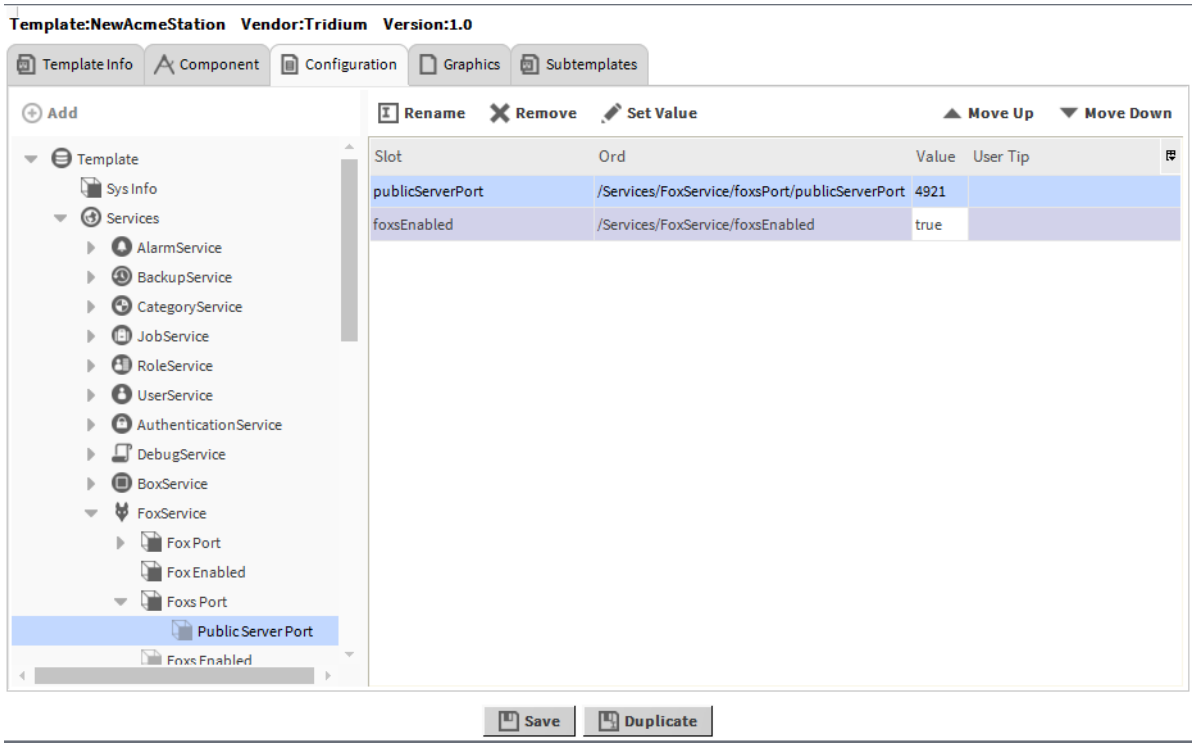
Step 1 To open the **Template View** , right-click on the station's **Config** node in the Nav tree, and select **Make Station Template**.

The **Template View** opens.



Step 2 Click each of the tabs to configure the station template as needed.

For example, to set up the template to configure the **`Foxs Port:`**, click the **Configuration** tab.



a. In the left pane of the **Configuration** tab, expand **Services→FoxService→Foxs Port**.

b. Double-click **`Public Server Port`**.

   The system adds it to the right pane.

c. Change the default value as needed and click **Set Value**.

   The value entered into the station template for the exposed property becomes the default value upon creating a new station. Any exposed component properties defined in the template appear as configurable parameters in the **New Station Wizard** when this template is used.

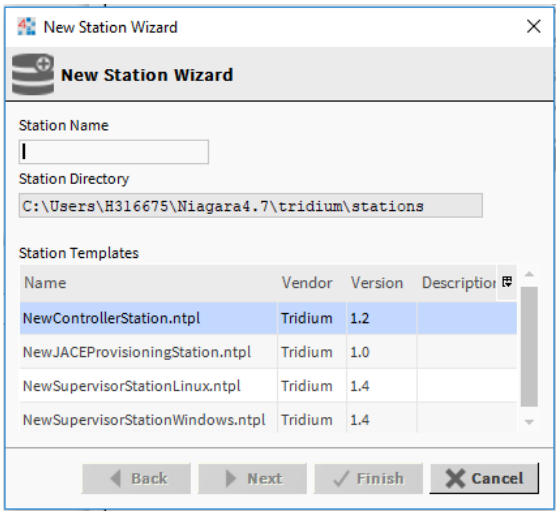
d. Click **Rename** and change the name to `Foxs Port`.

Step 3 On the **Subtemplates** tab, review and/or modify any deployed templates contained by the station, which are now sub-templates of the station template you are creating.

Step 4 When finished modifying the template, click **Save**.

The new station template is saved in the `~stationTemplates` sub-directory of your Workbench User Home.

Also, the station template is immediately available for inclusion in the Workbench **New Station Wizard**, where your template appears as a selectable option in the **Station Templates** table as shown here:



NOTE: It is not necessary to restart Workbench in order for the user-defined station template to appear in the **New Station** wizard. The file simply needs to be saved to the`~stationTemplates` directory.

More details on making a template are available in the *Niagara Templates Guide*.

Making a template module

A template module is a group of templates contained in a single `.jar` file that can be distributed as needed, either internally or to customers.

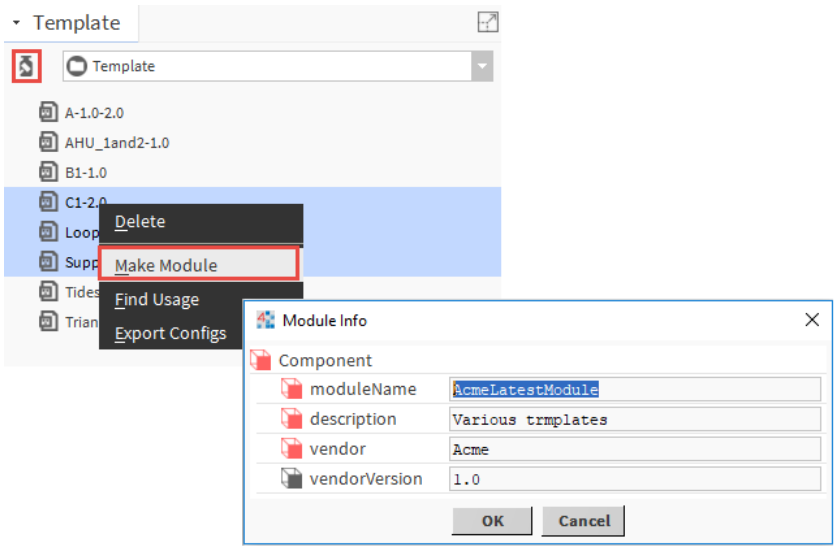
Prerequisites: One or more templates, which have been previously created and tested, exist. The **Template** side bar is open.

This procedure explains how to group one or more existing templates into a `.jar` file.

Step 1 In the **Template** side bar, expand the `Template` folder.

Step 2 Select one or more templates, right-click and select `Make Module` or click the (Make Module) icon in the **Template** sidebar header.

The **Module Info** window opens.



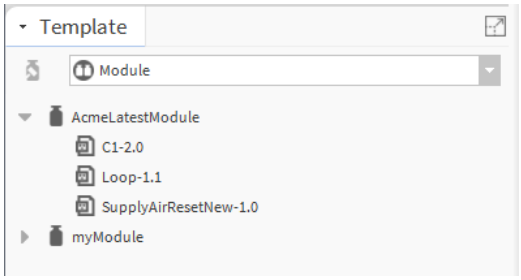
Step 3 Fill in these properties and click **OK**.

The system uses the **moduleName** as the `.jar` name as well as the node name in the **Template** side bar.

Step 4 To view templates in the module that you have created, do one of the following:

a. In the **Template** side bar, select the `Module` folder from the drop-down list.

Available template modules display in the **Template** side bar.



b. In the Nav tree, expand **My File System→Sys Home→modules** and navigate to your new `.jar` file.

NOTE: A template stored in a module cannot be edited. If opened, you see `ReadOnly` in the top left corner of the **Template View**. To make any changes you must first click **Save As** and save the template as a new template in the **templates** folder.

Chapter 3 Template deployment

Topics covered in this chapter

♦ Deploying a template via drag-and-drop
♦ Deploying a template via the Device Manager
♦ The provisioning of deployed templates

Template deployment is a multi-step copy operation. Once installed, the objects from the template become part of the runtime station and can be modified as needed. You can deploy a template by dragging the template from the **Template** sidebar onto the Nav tree, a **Wire Sheet**, or **Property Sheet** view, or by device discovery using **Device Manager** and adding a selected template.

NOTE: Be careful to always match a template to a device for which the template is designed. Do not associate just any template with any device.

NOTE: To avoid future problems, make sure that you tag and configure devices correctly before making a template and using it to configure other devices. Additionally, you must copy all relevant tag dictionaries (with the exception of the Niagara tag dictionaries) from the tag dictionaries Service to the tag dictionaries folder in your User Home file space before the template is first created (or edited), and during template deployment to be successful. Also, the system does not automatically apply subsequent template changes to the configured devices.

Deploying a template via drag-and-drop

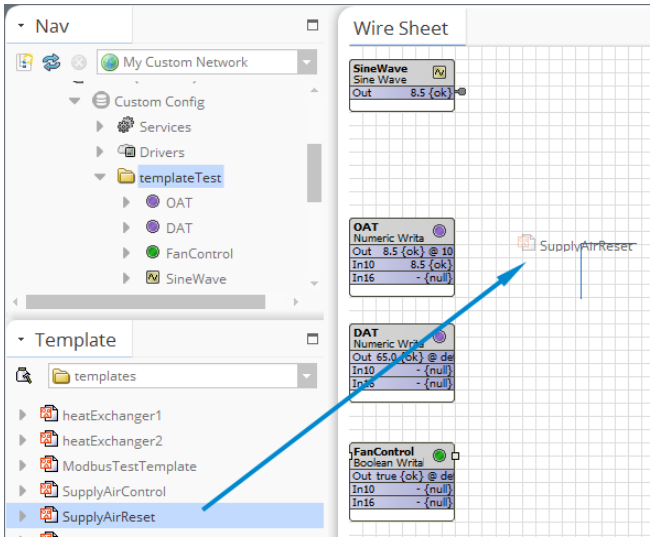
Deploy a template on a device by dragging from the **Template** side bar onto the device's component space. This is similar to dragging and dropping components from a palette.

Prerequisites: An existing template is available.

This procedure demonstrates the deployment steps using a template for a Supply Air Reset component.

Step 1 Right-click on the device and open the wire sheet view where you want to deploy the template.

Step 2 In the **Template** sidebar, click on the template and drag it to the wire sheet view.



The template **Name** window appears.

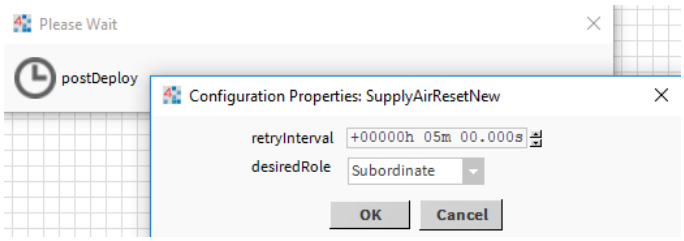
Step 3 Change the template name as needed and click **OK**.

NOTE: Clicking **Cancel** on any of these windows aborts the deployment process and removes the installed devices. It does not remove any installed Px or image files.

The **Please Wait** window appears, that indicates which files are being transferred to the target station. Additionally, the window updates with information as it does NEQL queries for the inputs, outputs, and relations.

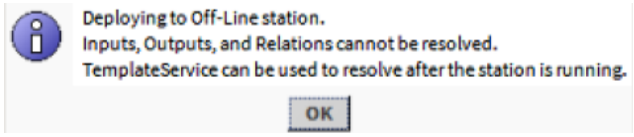
NOTE: If the template file contains a **`Password`** with the installed value `BPassword.DEFAULT`, a window displays alerting the user of this fact. The TemplateService and **Template Manager** view can be used to later locate the password and assign a value.

The **Configuration Properties** window appears.



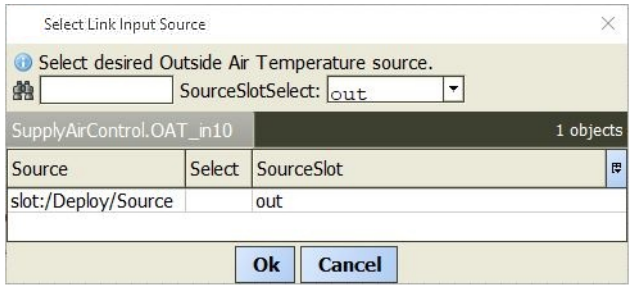
Step 4 If you wish, edit the configuration properties and click **OK**.

NOTE: If deploying a template to an off-line station the following **Info** window appears. At this point deployment is finished. Once the station is running, you can access the **TemplateService** and **Template Manager** view to resolve the unbound inputs, outputs, and relations.



For each template input defined with bindHints that return one or more results, the **Select Link Input Source** window appears.

Step 5 In the **Select Link Input Source** window, approve or configure input sources (repeat as needed to approve or configure all template input links).



Note that a template input must be linked from only one source. Also, input source points can be located anywhere in the station.

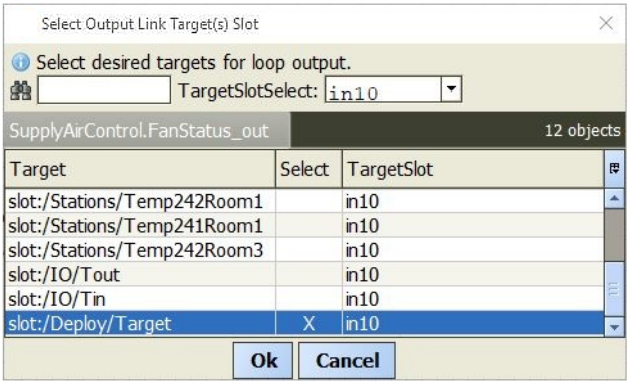
- If an input resolves to a single source it is automatically selected for you, click **OK** to accept the source.
- If the input resolves to multiple possible sources, you must select one by clicking the **Select** cell of the desired input source, and click **OK**.
  - If a userTip is defined, it displays at the top of the window. For example, in the above image the userTip, "`Select desired Outside Air Temperature source`", is visible.
  - If the list of sources is long, you can sort the list by clicking on the**Source** column heading. You can also filter the list by typing in the search text field.

- If no sources are detected for an input, a popup appears indicating that fact. Click **OK** to proceed.

  NOTE: In this situation, you may need to create an input source point(s) and tag it appropriately so that the input link resolves correctly. You can either cancel deployment in order to create and tag any undetected input source points and then redeploy the template, or just proceed and after deployment create and tag any missing input sources and then resolve those inputs by accessing the Template Service.

For each template output defined with bindHints that return one or more results, the **Select Output Link Target(s) Slot** window appears.

Step 6 In the **Select Output Link Target(s) Slot** window, click the **TargetSlotSelect** drop-down or double-click the **TargetSlot** cell, click to select a slot and then click **OK** (repeat as needed to configure all output links).



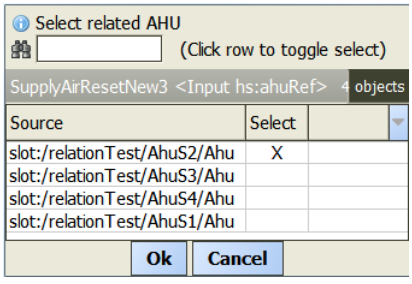
If multiple points are detected, each point will be listed and the target slot selection initialized based on the `targetSlotHint` tag value.

If no `targetSlotHint` inputs are linkable, the `----` (do not link) option will be selected. The target slot choices will be limited to the target point's linkable input slots, i.e. if the `in10` input is already linked it will not appear in the choice list. If you do not wish to link to a matching point select the `----` option.

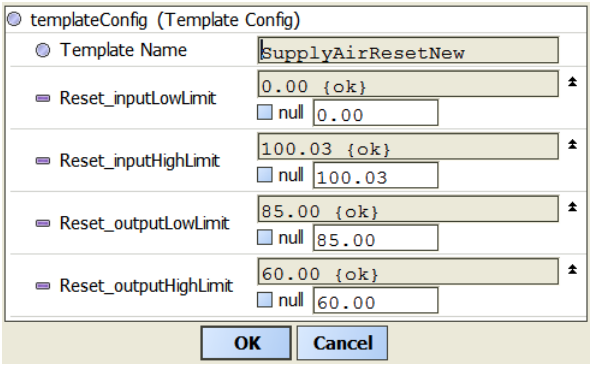
NOTE: A template output can be linked to more than one target.

If the template is setup with one or more component relations, the **Select Related Component** window appears.

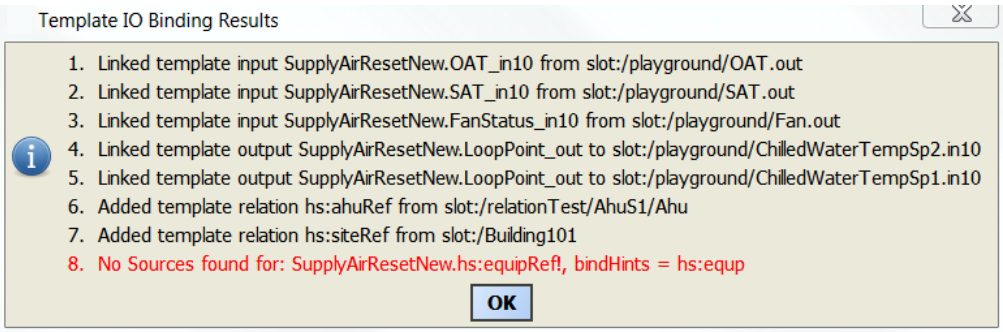
Step 7 In the **Select Related Component** window, approve or configure component relations (repeat as needed to approve or configure all template component relations).



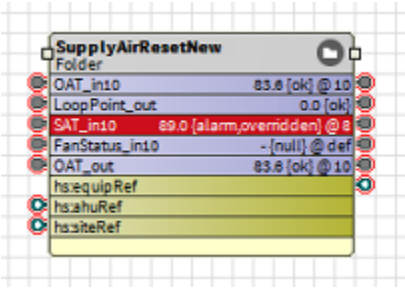
Step 8 A Configuration Properties window appears. Edit the configuration properties as desired.



Step 9 Once all inputs and outputs have been processed the **Template IO Binding Results** window appears indicating the linked inputs, outputs, and added relations. Click **OK**.



Template deployment is complete. The new fully configured component is visible on the device's wire sheet.



Deploying a template via the Device Manager

Device Manager provides support for deploying templates whose base component is a Device component. If a driver supports discovery and matching operations then these mechanisms allow template deployment as well.

Prerequisites:

- An existing template
- A network with one or more devices for which the template is designed

Step 1 Navigate to the **Station→Drivers→Network** container (for example, BacnetNetwork) and double-click on the network to open the **Device Manager** view.

The **Database** table contains all devices that have already been configured.

Step 2 Click **Discover**.

The **Discovered** pane appears listing devices discovered on the driver's network, as candidates for inclusion in the station database.

Step 3 Click (**Dev Template Mode** button) on the tool bar, or on the menu bar click **Manager→Dev Template Mode**.

This opens a **Templates** pane in the upper portion of the **Device Manager** view, displaying a list of templates that are valid for the network.

Step 4 Select one or more devices (of the same type) from the **Discovered** pane.

The **Templates** pane changes to display only those templates that are valid for the selected discovered device(s).

Step 5 In the **Templates** pane, select the template you wish to add for the selected discovered device(s).

NOTE: Be careful to always match a template to a device for which the template is designed. Do not associate just any template with any device, even if it is valid for a discovered device.

Step 6 Click **Add** to start the deployment process. At this point the following occurs:

- The template copies an instance of the template bog file for each selected discovered device into the station. The devices are disabled at this time.
- The template copies a single instance of any Px and image files into the station.
- For each deployed instance, the template queries for the input, output and relation source, target, and related choices.

  NOTE: Clicking **Cancel** on any of the following selection windows aborts the deployment process and removes the installed devices. It does not remove any installed Px or image files.

  – For each input, the template prompts the user to select the source. This window have the option to reuse the selection for this input for all the remaining deployed devices. It does not prompt for this same input for other devices.

  – For each output, the template prompts the user to select one or more targets. This window does not provide the option to reuse the selection.

  – For each inbound relation, the template prompts the user to select one or more related sources. This window provides the option to reuse the selection for the other deployed devices.

  – For each outbound relation, the template prompts the user to select one or more related targets. This window provides the option to reuse the selection for the other deployed devices.

- The selected template is deployed for each of the selected discovered devices. A **Template IO Binding Results** window appears for each of the selected devices. Click **OK** to close.
- For each installed device template, the template attempts to match to the corresponding selected discovered device, and enable the device.

The **Edit** window appears. A list of the added devices displays in a table at the top of the window with a tabbed pane below. The **Template** tab allows you to edit the template configuration properties. The **Properties** tab allows you to edit the typical properties for this device.

Step 7 In the **Edit** window, select each device in the table and make desired edits (on the tabs) as needed:

- Update any properties that appear on the **Template** tab with the values that are unique for the current device.
- And if not already applied, you can select the **Properties** tab, create a unique name for each device or change its MAC address.

Step 8 After editing each device listed in the upper table, click **OK**.

Template deployment is complete.

NOTE: Unmatched templates can be added to the station by simply double-clicking on a template or dragging and dropping a template on the **Database** table. The user is prompted for template input sources and template configuration changes, as described above. In this case, the device can be manually addressed or matched to a device at a later time.

For drivers that do not support discovery
For drivers that do not support discovery, you can still use the network **Device Manager** view to deploy a template.

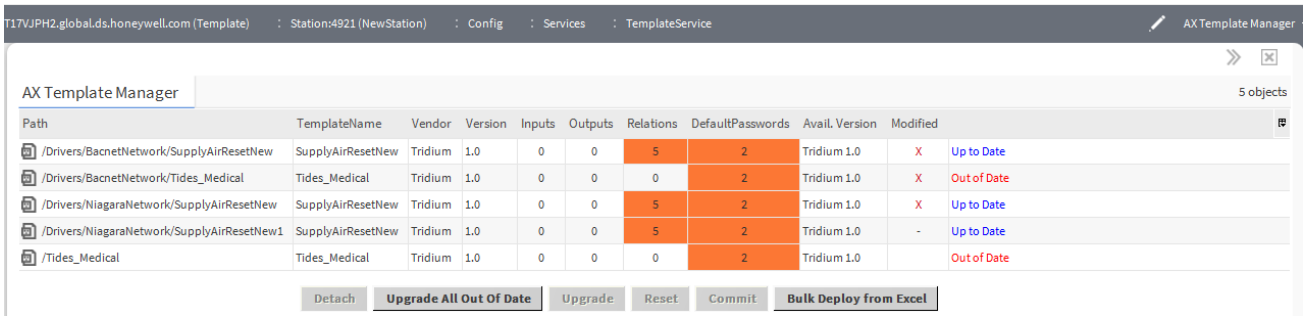
1. Click the **Dev Template Mode** button on the toolbar, and in the resulting **Template** pane select the desired template.
2. Click **New** and add one or more instances of a device using the selected template.

The provisioning of deployed templates

In Niagara 4.3 and later, the **Template Manager** view supports updating one or more deployed templates in a single action. The functionality facilitates upgrading previously deployed templates by monitoring the version and status of deployed templates, and allowing you to make modifications to templates. On completion of modifications to a source template, you can apply the changes either to all out-of-date deployed template instances or to a select number of deployed template instances.

The **Template Manager** view lists all deployed templates and subtemplates in the station. Subtemplates appear indented beneath their parent template. This view lets you detach, upgrade/downgrade, reset, and commit, or simply review all deployed templates in a station.

The status of deployed templates is indicated in the untitled column in right-side of the view. This status modifies the choices of enabled provisioning commands (visible as either dimmed or activated buttons) at the bottom of the view.



Upgrading a deployed template

This procedure documents how to upgrade a deployed template in the station.

Prerequisites:

- One (or more) existing deployed templates with the "Out of Date" status displayed in the **Template Manager** view.

CAUTION: Note that upgrading deployed templates wipes-out any local changes you have made to those objects.

Step 1 Click to select a single row that shows a template is "Out of Date" and click **Upgrade**.

This flags the selected template to be upgraded.

Optionally, if multiple templates appear "Out of Date" you can click **Upgrade All Out Of Date** to flag all out of date templates to be upgraded.

Step 2 Click **Commit** to initiate deployment.

NOTE: The deployment process completely removes the existing template and installs a newer version.

the **Commit Template** window opens.

Step 3 In the **Commit Template** window, click **Yes** to continue.

NOTE: The Upgrade, Downgrade, and Redeploy procedures all use the same deploy process to complete provisioning.

Upon completion of upgrading the deployed template the **Template Manager** view shows the template's status as, "Up to Date".

Downgrading a deployed template

This procedure documents how to downgrade a deployed template in the station.

Prerequisites:

- One (or more) existing deployed templates with the "Older Version Available" status displayed in the **Template Manager** view.

Step 1 Click to select a single row that shows a template has an "Older Version Available " and click **Downgrade**.

This flags the selected template to be downgraded.

Step 2 Click **Commit** to initiate the deploy process.

Step 3 In the **Commit Template** window, click **Yes** to proceed.

NOTE: The deployment process completely removes the existing template and installs a newer version.

Step 4 In the **Commit Template** window, click **Yes** to continue.

NOTE: The Upgrade, Downgrade, and Redeploy procedures all use the same deployment process to complete provisioning.

Upon completion of downgrading the deployed template, the **Template Manager** view shows the template's status as "Out of Date."

Redeploying a deployed template

This procedure documents how to redeploy a deployed template in the station.

Prerequisites:

- One (or more) existing deployed templates with the "Up to Date" status displayed in the **Template Manager** view.

Step 1 In the **Template Manager** view, click to select a row that shows the template is "Up to Date" and click **Redeploy**.

This flags the selected template to be redeployed.

Step 2 Click **Commit** to initiate deployment.

NOTE: The deployment process completely removes the existing template and installs a newer version.

Step 3 In the **Commit Template** window, click **Yes** to continue.

NOTE: The Upgrade, Downgrade, and Redeploy procedures all use the same deploy process to complete provisioning.

Upon completion of redeploying the template the **Template Manager** view shows the template's status as, "Up to Date".

Detaching a deployed template

The detach procedure may be used when you need to make modifications to a deployed template and you have no intention of upgrading its previous deployments.

Prerequisites:

- An existing top-level deployed template is available.
- The **Template Manager** view is open.

CAUTION: When you detach a deployed template it is no longer handled as a deployed template. This means that the template cannot be upgraded in the future.

Step 1 In the **Template Manager** view, select the row of a top-level template.

The **Detach** button is enabled.

Step 2 Click **Detach**.

Step 3 In the **Detach** confirmation window, click the **Remove Composite Slots** checkbox and click **Yes** to confirm that you wish to continue.

The deployed template is detached.

Chapter 4 Application Template

Topics covered in this chapter

♦ Application Template vs. Station Template
♦ Application Template vs. Component (Device) Template
♦ Existing file and component structures for templates
♦ Creating an application template
♦ Installing an application template
♦ Upgrading an application template
♦ Optional components in application templates

In Niagara 4.7 and later, the Application Template is available.

Application templates are a mechanism to deploy an entire station application to a running station. They are the primary means of sharing solutions that are built or Edge devices. In concept, application templates can be used wherever a whole station needs to be replicated and installed to running stations.

The information provided here compares Application templates to Station templates and to component templates, explaining the similarities and differences between each. Also, this section covers how to create and install an Application template.

Application Template vs. Station Template

This topic documents the similarities and differences between application template and station template.

Similarities

- Station templates and application templates define the full contents of a station and encapsulate the graphics files and subtemplates that are used by the station.
- Both templates can be edited in Workbench that includes, defining of configuration properties for the template.

Differences

Whereas station templates are consumed only by Workbench for the creation of a new station; application templates can be installed to a running station, replacing most of the station contents. Station templates cannot be upgraded, but application installations can be upgraded to a newer version of the template.

- Station templates are installed during the creation of a new station; however application templates can be installed to a running station, replacing most of the station contents.
- Station templates can not be upgraded, however application templates can be upgraded to a newer version of the template.

Application Template vs. Component (Device) Template

The term component template also represents the device template.

Similarities

- Application templates and component templates are installed (deployed) to a station in a similar way.
- Application templates and component templates can be edited and configuration properties can be defined.
- When deploying component templates or installing application templates, the defined configuration properties can be set.

- Application templates and component templates can contain graphics and subtemplates.
- The subtemplates of both application templates and component templates must be component templates.

Differences

- Component templates can define inputs, outputs, and references; these options do not exist for application templates.
- The target for a component template can be any container, but the user must select a container that is appropriate for the components of the template. In contrast, an application template can only be installed to the root Config folder of the station.
- The root component of a component template is given a user-assigned name when it is deployed. The names of components installed by an application template are fixed.
- Deploying a component template always adds a new component tree to a path that did not exist before; the name may be modified during the deployment to make its path unique. For an application template, existing components in the station will be deleted first to ensure that no functionality from the old application remains; also it confirms that there will be no naming conflicts in the resulting station.
- Multiple deployments of a component template can be created in a station; an application template has one installation in a station and there is only one application in the station
- In Niagara 4.7, bulk deployment of application templates is not available. They can be deployed to a single station manually, or can be deployed to multiple stations by using provisioning.
- The bulk deploy spreadsheet for a component template includes columns for parent path, component name, display name, and location; none of these columns are relevant for application templates. (Not applicable for Niagara 4.7)
- Bulk deploy can create many uniquely-named instances of a component template in the target. Since a target can only have one application template, bulk deploy for these is perhaps a contradiction, but the spreadsheet is a way to uniquely configure the one instance that can be created. (Not applicable for Niagara 4.7)

Existing file and component structures for templates

Station templates and component templates use the same file structure.

The file is zip-compressed, has a NTPL file extension for Niagara template, and encapsulates at least two internal files:

- template.bog
- template-manifest.xml.

Graphics and other internal files may be included as needed to complete the template definition.

For a station, the embedded BOG file will be the station BOG; for a component template, it will be the component tree that was selected for the template. In either case, the resulting component tree will include a templateConfig component of type BTemplateConfig that captures the additional template configuration and metadata. The templateConfig component is a direct child of the template root component; it is added when the template is first created.

Within a station, deployed component templates are identified by the presence of the templateConfig component as a direct child of the template root component. Once a deployed template is found, it may be further differentiated as a device template by examining the component type of the root component.

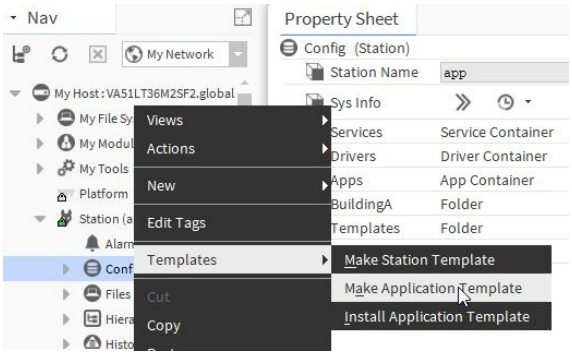
Application templates have a file extension of NAPL. This distinguishes them from other template types, and the unique file extension enables these files to be used differently throughout the Niagara system. Application template files are stored in a separate folder from component template files, and they are not available from the **Template** sidebar in Workbench. Deployed application templates place the templateConfig component under the **Application Service** in the **Services** container.

Creating an application template

Application templates are created in the Workbench that connects to a running station with a completed set of component logic, device networks, and graphics.

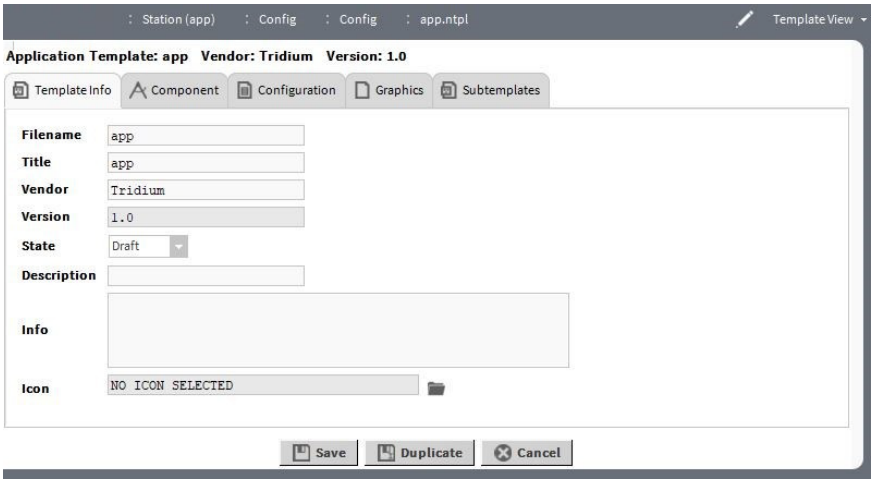
NOTE: To avoid future problems, make sure that you tag and configure devices correctly before making a template and using it to configure other devices. Additionally, you must copy all relevant tag dictionaries (with the exception of the Niagara tag dictionaries) from the tag dictionaries Service to the tag dictionaries folder in your User Home file space before the template is first created (or edited), and during template deployment to be successful. Also, the system does not automatically apply subsequent template changes to the configured devices.

Step 1 Expand **Station→Config** in the Nav tree and right-click the **Config**.



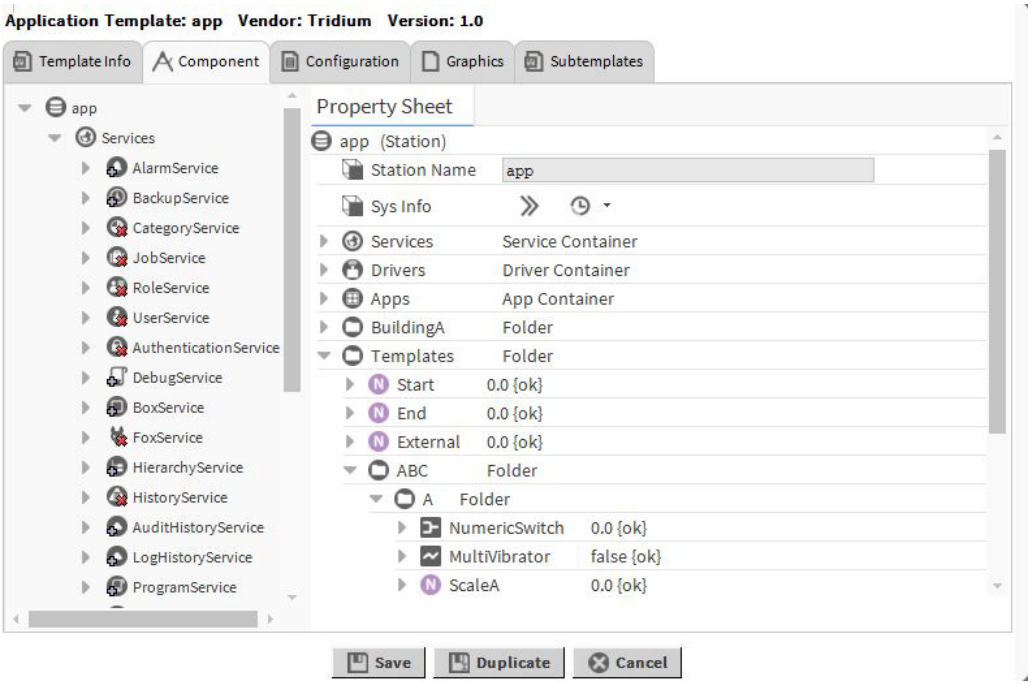
Step 2 Select **Templates→Make Application Template** from the option.

A **Template View** opens.



NOTE: This view is the same for all template types (Application, Device, Component), except that the Application Template does not have inputs, outputs, or relations

Step 3 Click each of the tabs to configure the application template as needed.



NOTE: The **Component** tab includes the components for the entire station source for the template. Template components can be added, modified, and deleted in this view. The Nav Tree for application templates show component icons with badges to indicate the deployment status of the component. Certain components are contained in the template but not deployed to the station during the installation.

In Niagara 4.7 and later, these are restricted to several critical **Services** and the NiagaraNetwork container under the **Driver**. These will be displayed with the red "+" badge. Components that will be deployed are displayed with the gray "+" badge. Container components that have both deployable and non-deployable child components do not have a badge.

Step 4 To generate the template file, click **Save**.

The file is saved to the `applicationTemplates` folder in the user home. If the file name already exists, Workbench prompts for confirmation before overwriting the existing file. The file can be reopened for editing by finding the file in the Nav Tree sidebar and double-clicking on the template NAPL file.

Application templates are not available in the **Template** sidebar due to the destructive nature of the application template deployment. The Version and State of the template can be modified, if changes are made. The version is maintained on the deployed templates in the Template Config component in the **Application Service**.

Installing an application template

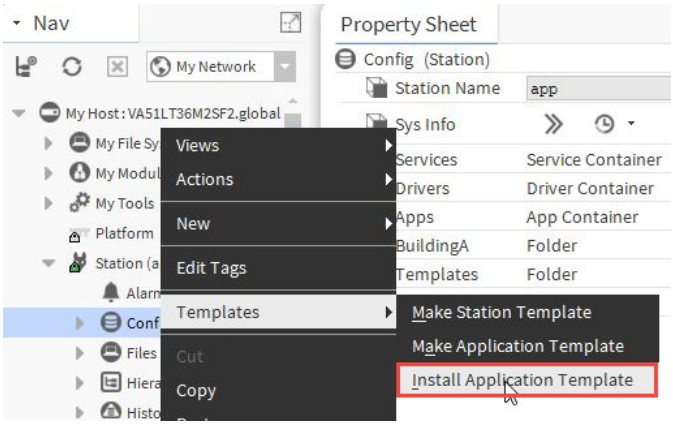
This topic describes how to install an application template on a target station.

Prerequisites:

- Application templates will not install to a station where the platform does not have all required modules installed, including any modules containing required PX pages or images.
- Target stations must have the Job and Template services already installed.
- The WB (-wb) runtime profile is required to install templates.

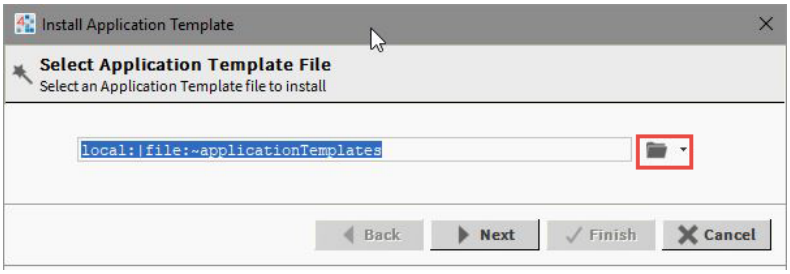
CAUTION: Be aware that this is a destructive operation. Existing station components will be deleted.

Step 1 Expand **Station→Config** in the Nav tree and right-click the **Config**.



Step 2 Select **Templates→Install Application Template** from the option.

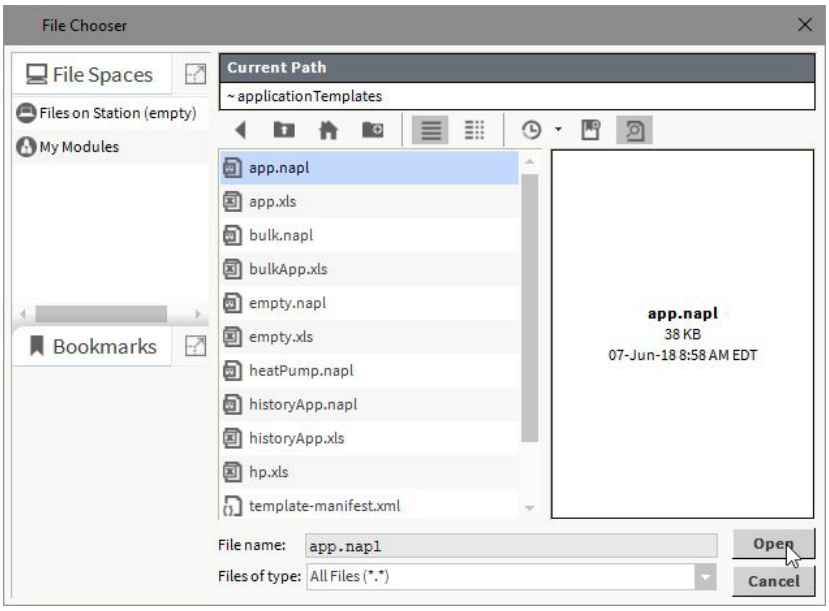
An **Install Application Template** window opens.



Step 3 Click the Folder icon and click the `File Chooser` option.

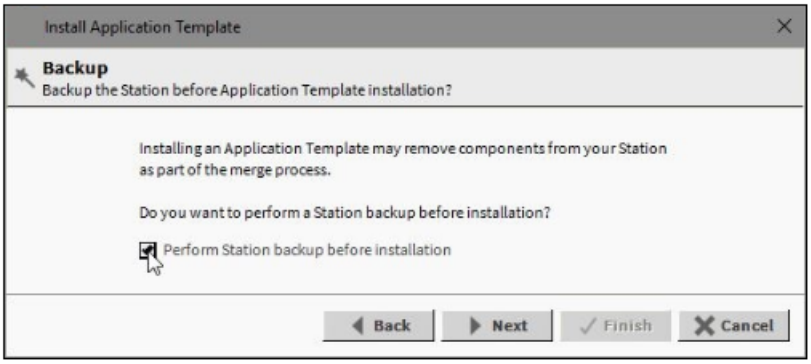
A **File Chooser** window opens.

Step 4 Click the application template file you want to install and click **Open**.



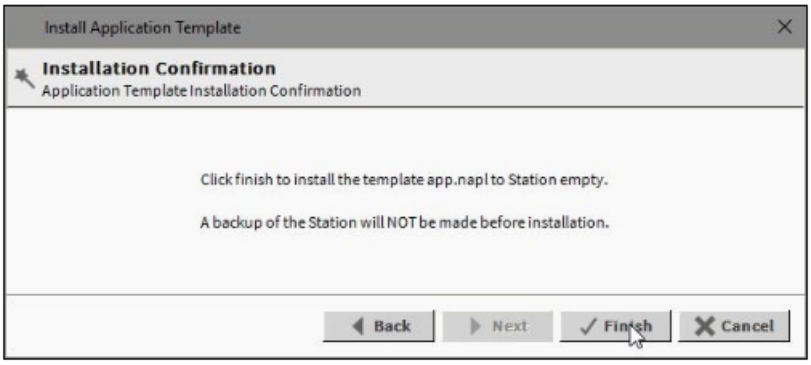
The **File Chooser** closes and a second **Install Application Template** window opens. The option to create a backup of the existing station before the application template is installed is selected by default.

Step 5 If you do not want a backup, deselect the **Perform Station backup...** checkbox to remove it. Click **Next** to proceed.

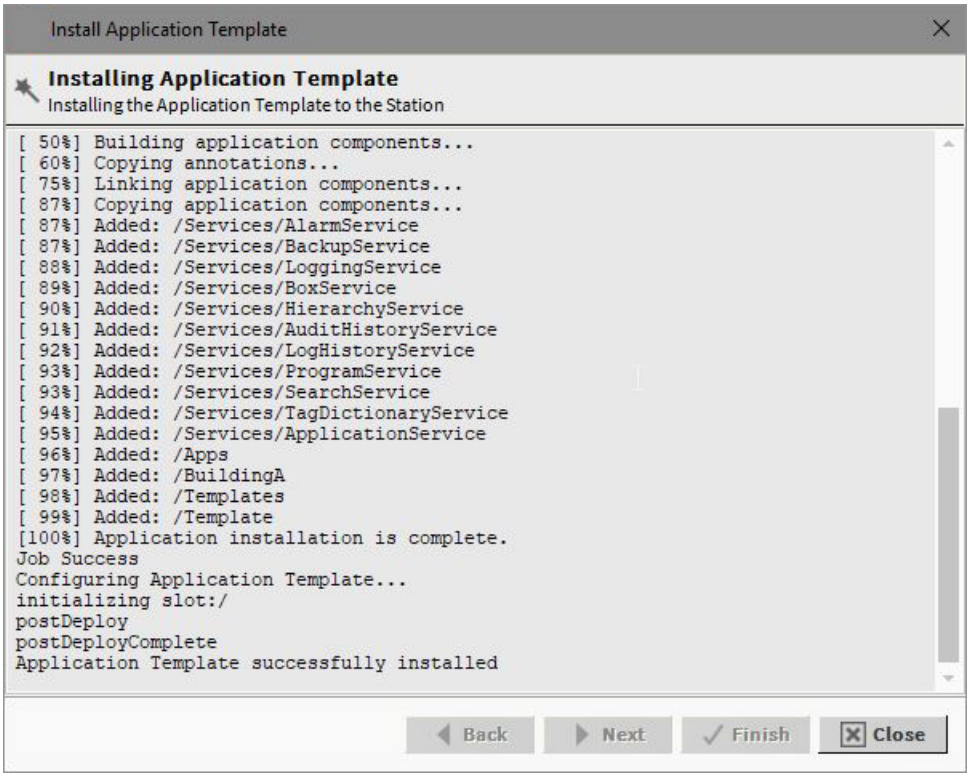


The third **Install Application Template** window opens.

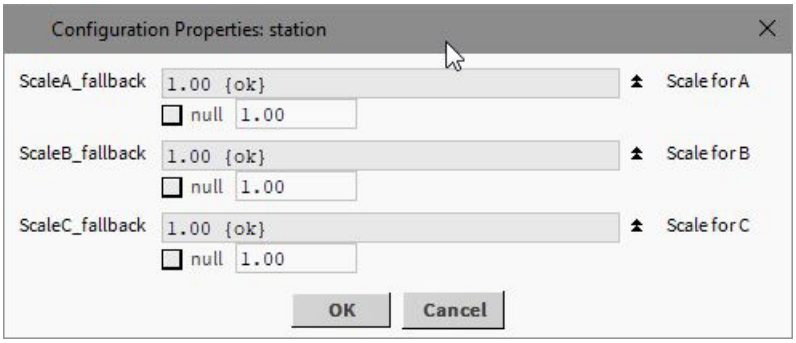
Step 6 Click **Finish** to confirm the installation.



The fourth **Install Application Template** window opens, showing updates of the deployment progress.



Step 7 A **Configuration Properties station** window opens to allow you to change any default configuration value for the application template and click **OK** to continue.



Step 8 Once the installation is complete, click **Close** in the progress window.

The installation runs via the Job Service on the target station, so the installation log can be reviewed through the **Job Service Manager** view on the Job Service.

NOTE: The current application template implementation does not provide advanced template version management. There is no version or change tracking for installed application templates. In Niagara 4.9 and later, there is added support for application template upgrade, but to accomplish the same in prior releases the new template version must be re-installed.

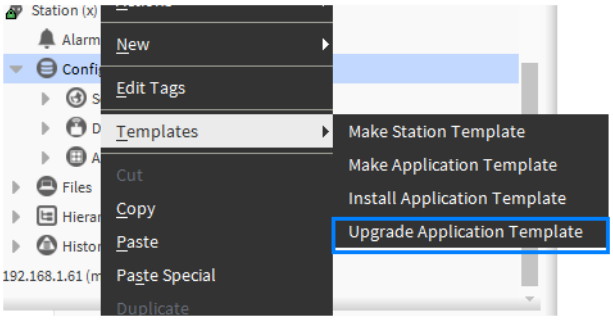
Upgrading an application template

Niagara 4.9 (and later), supports application template upgrade. This procedure describes how to upgrade an application template.

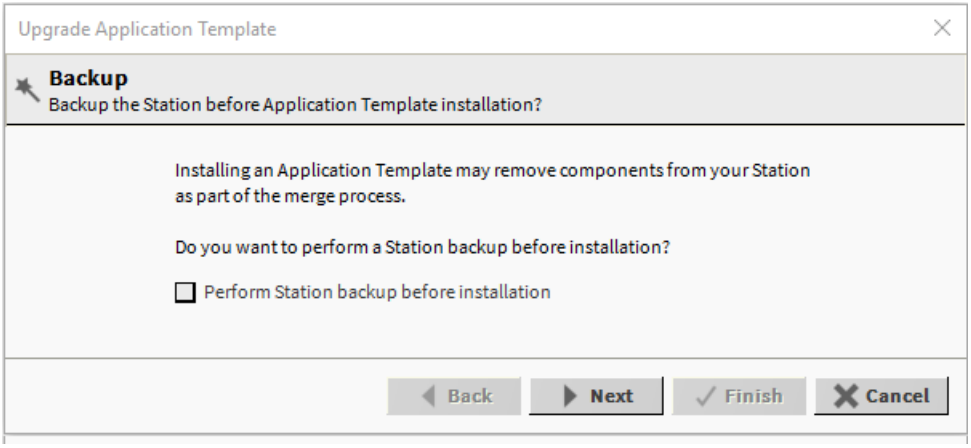
Prerequisites:

When a station is running an application template and the Workbench user home applicationTemplates directory contains a newer version of the same template (as indicated by a higher version number assigned when it was last saved), the **station→Config→Templates** sub-menu will include the **Upgrade Application Template** menu option.

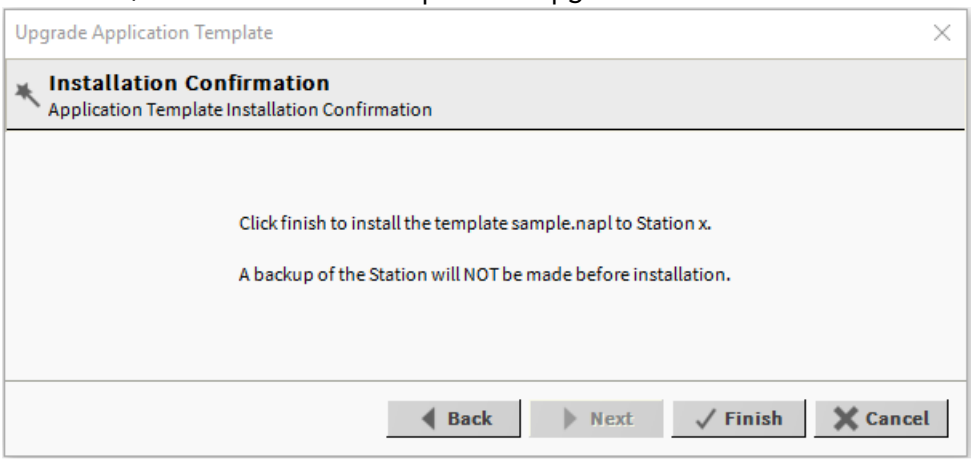
Step 1 In the Nav Tree, click **station→Config→Templates→Upgrade Application Template**.



This launches the **Upgrade Application Template** wizard, a simplified version of the application template install wizard, that includes an optional station backup step, and confirmation step.

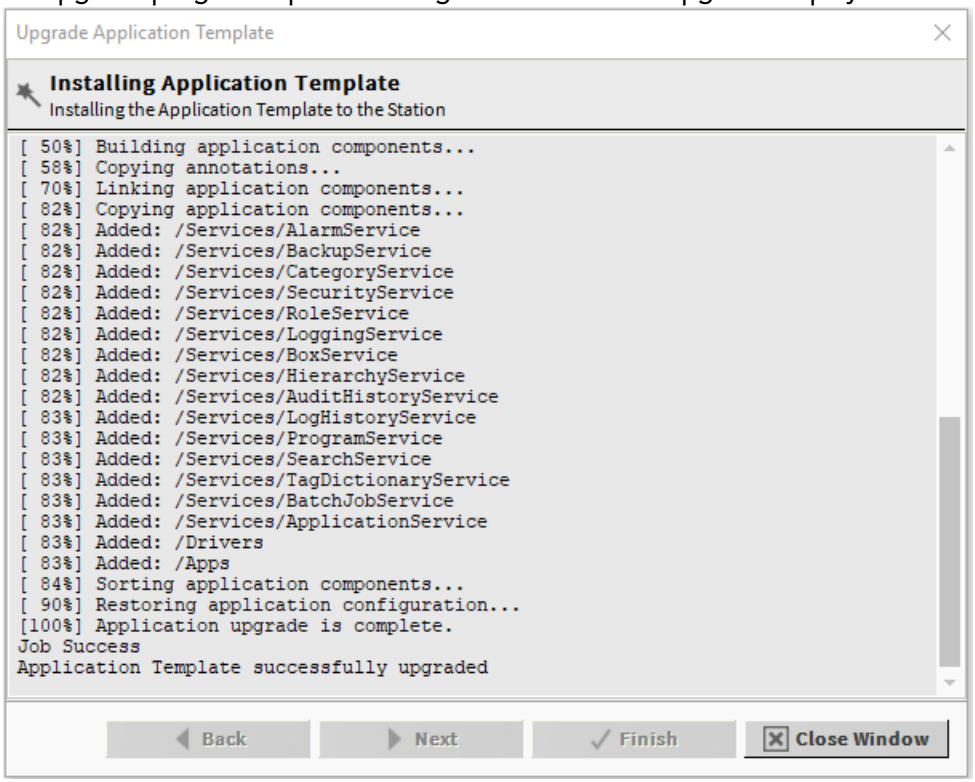


Step 2 Click **Next**, and click **Finish** to complete the upgrade installation.



CAUTION: The application template configuration will be preserved by the upgrade, but ad hoc edits made outside of the template configuration may be lost during the upgrade.

An upgrade progress report detailing the results of the upgrade displays in the window.



NOTE: Alternatively, upgrading application templates can also be accomplished through provisioning. For more details, see the latest version of the *Niagara Provisioning Guide.*

NOTE: If a component of a template is accidentally deleted, upgrading the template to a new version will restore the deleted component unless that component is marked as optional for the new template version.

Optional components in application templates

In Niagara 4.9 and later, application templates support the declaration and installation of optional components.

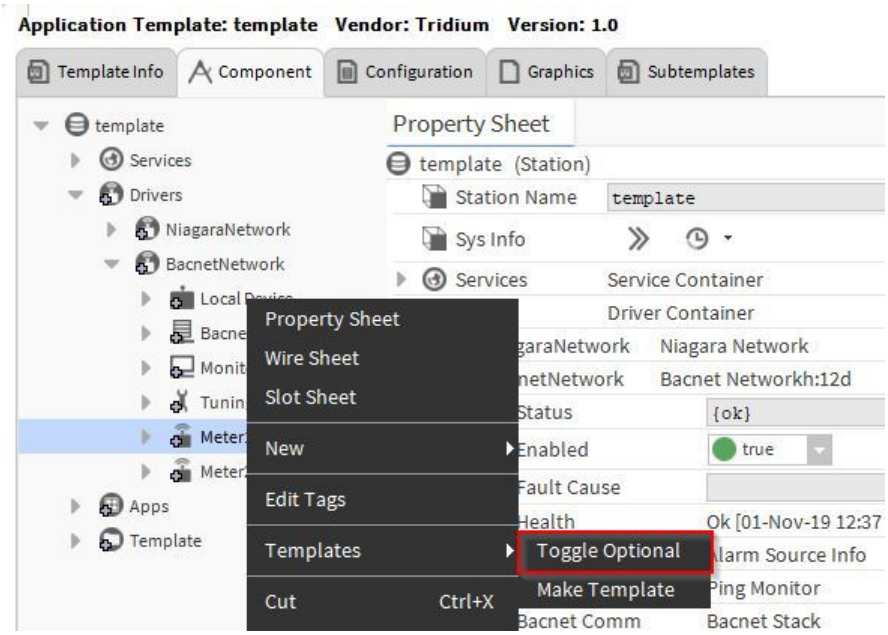
Optional components must be declared using the template editor. Only a Device or Device Folder may be declared optional. Optional components can be individually selected for installation during the application template installation workflow.

Establishing optional components

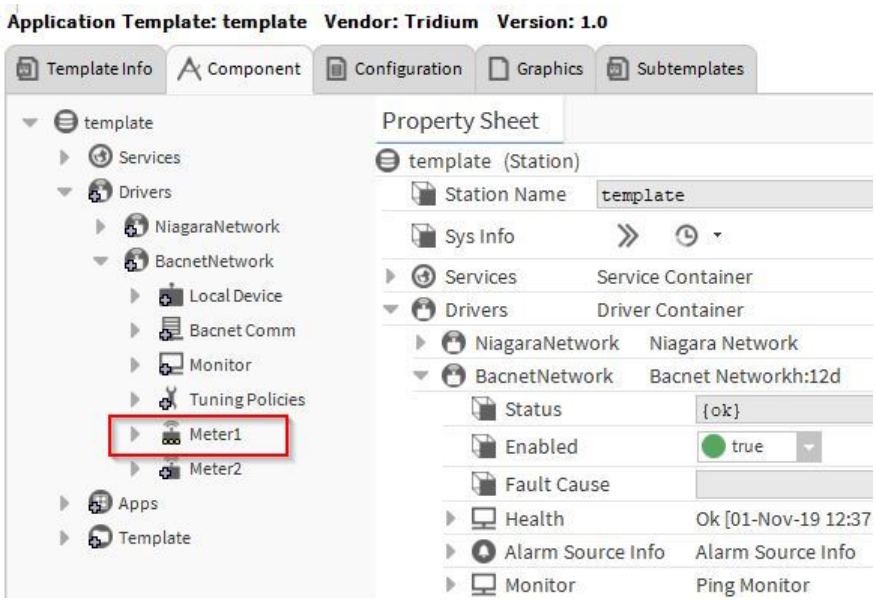
Optional components are established in the **Component** tab of the **Template View**.

Step 1 In the **Nav Tree**, navigate to a Device or Device Folder.

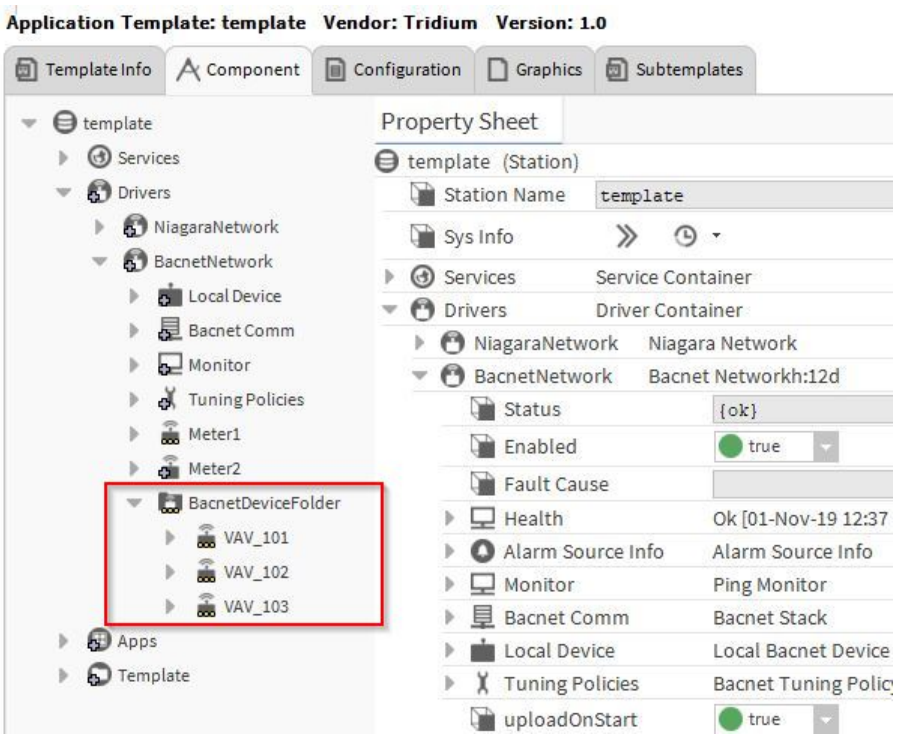
Step 2 Right-click on the Device or Device Folder and click **Templates→Toggle Optional**.



Notice that the badge for the Device or Device Folder changes from a plus (+) to an ellipsis (. . .).



If the selected component was a Device Folder, all devices within that folder will be marked as Optional with the ellipsis (. . .) badge.

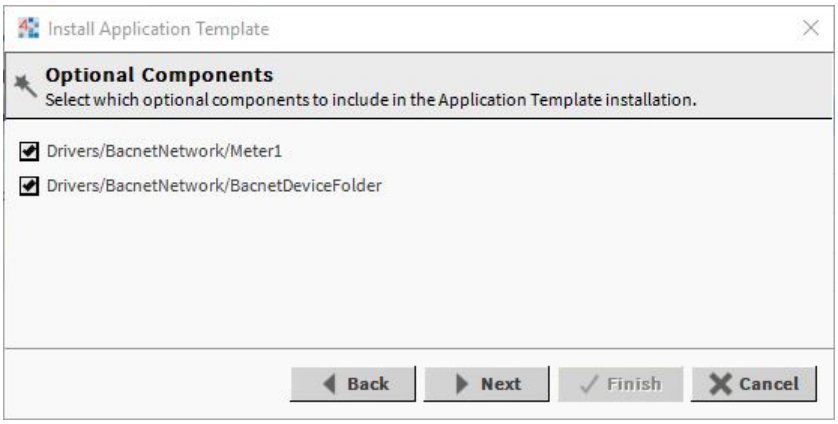


NOTE: Regarding the impact of upgrading an application template or updating template configuration to optional components, it is important to note that neither can change the selection of optional components installed. In order to change the selection of optional components, the template must be re-installed.

Installing optional components

Optional components are installed as part of the regular application template installation process described earlier.

If the template contains optional components, an additional installation wizard step dialog will be displayed to prompt for the optional components to be installed.



All optional components are pre-selected by default.

Step 1 Uncheck any Device or Device Folder instances to disable their installation.

Step 2 Select **Back**, **Next**, or **Cancel** as desired to proceed.

Chapter 5 Template bulk deployment

Topics covered in this chapter

- ♦ Exporting the template spreadsheet
- ♦ Editing a spreadsheet
- ♦ About the exported spreadsheet
- ♦ Deploying bulk templates
- ♦ Exporting the application template spreadsheet
- ♦ Editing the application template spreadsheet
- ♦ Installing bulk application templates
- ♦ Updating template configuration
- ♦ Optional components in application templates

Starting in Niagara 4.7, a single operation can deploy multiple instances of one or more templates to a station.

Template bulk deployment differs from bulk provisioning in that provisioning configures multiple stations in a single operation, whereas template bulk deployment configures a single station with multiple instances of one or more templates. This operation loads information into the station for each device or other entity the station supports.

NOTE: Template bulk deployment does not support automatic bulk deployment of sub-templates.

NOTE: Enhancements to template bulk deployment, in Niagara 4.8 and later, include added support for the following:

- Excel `*.xlsx` format
- Password configuration in templates
- Password encryption on template exports

NOTE: To avoid future problems, make sure that you tag and configure devices correctly before making a template and using it to configure other devices. Additionally, you must copy all relevant tag dictionaries (with the exception of the Niagara tag dictionaries) from the tag dictionaries Service to the tag dictionaries folder in your User Home file space before the template is first created (or edited), and during template deployment to be successful. Also, the system does not automatically apply subsequent template changes to the configured devices.

Prerequisite

Bulk deployment assumes that the one or more templates to deploy exist.

Overview

The bulk deployment process involves these primary steps:

1. Select the template(s) to include in the deployment and create an Excel file that contains a worksheet for each template.
2. For each worksheet, fill in a row with the unique information required by each station device.
3. Select this Excel file to use in the deployment, and execute the deployment job. The job starts with the first worksheet in the Excel spreadsheet, extracts configuration data from each row, and configures each device in the target station. The job moves from left to right through the worksheets. Data dependencies must be located in the worksheets ahead of when they are needed.

Template bulk deployment does not work when a station is offline.

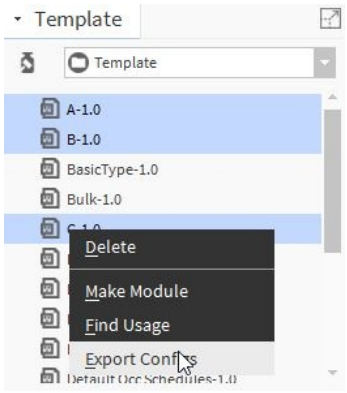
Exporting the template spreadsheet

Exporting template configuration properties to an Excel spreadsheet assembles in a single file the information to be deployed to each station. You may create and deploy more than one Excel file/template file set to complete an entire application installation or include multiple templates as part of the single Excel file. The export function generates a separate Excel worksheet tab for each template.

Prerequisites: The **Template** sidebar is open with the list of templates. The templates to include in the bulk deployment exist.

NOTE: In Niagara 4.7 and later, there is added support for configuring String Tag entries in the spreadsheet.

Step 1 Open the **`Template`** sidebar.

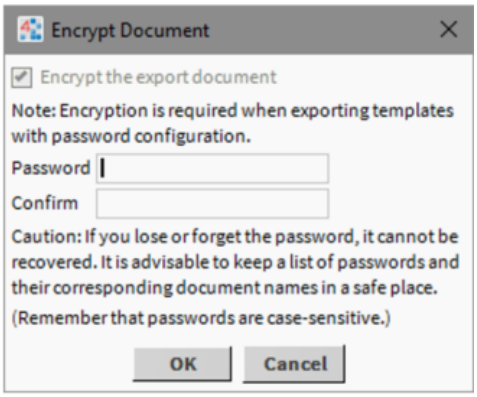


Step 2 Select one or more templates, right-click, and click **Export Config**.

The **File Chooser** window opens.

Step 3 Use the **File Chooser** to select a destination and name (extension `*.xls` or `*.xlsx`) for the exported file and click **Save**.

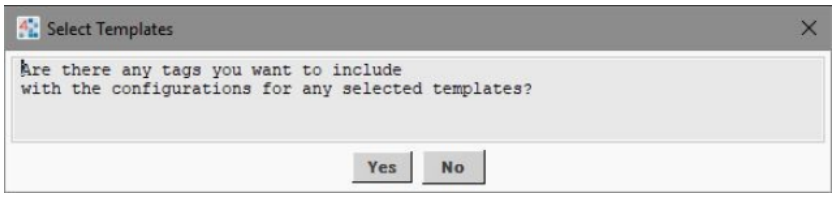
An **Encrypt Document** window opens. This document encryption is generally optional, but if the template contains password elements, the exported configurations must be encrypted.



If encryption is desired, enter and confirm the password for unlocking the exported file.

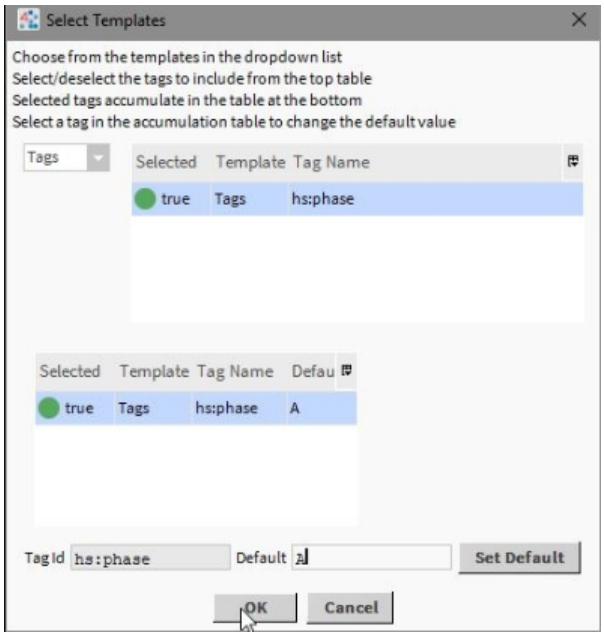
NOTE: An encrypted file cannot be saved in the `*.xls` format. A warning popup window will be displayed if this is attempted. Use the`*.xlsx` format instead.

Step 4 A **Select Templates** window opens.



Step 5 Click **Yes** to include string tags with the templates (no other tags, such as marker tags, can be included), or **No** to skip.

Step 6 If you click **Yes**, another **Select Templates** window opens. Select a template from the drop-down list (upper left).



The system populates the table to the right of the list with the set of string tags that are attached to any component in the chosen template. These are the tags that are available for export. The **`Selected`** indicator for all tags is initially set to `false`.

Step 7 To include a string tag, click its row in the table.

The **`Selected`** indicator changes to `true`.

The table at the bottom of the window contains the accumulated set of tags for export as well as the default value of the tag, which initially is blank.

Step 8 To change the **`Default`** value, select the row in the bottom table.

The system displays the **`Tag Id`** in the text property at the bottom of the window.

Step 9 Type in a value for the **`Default`** property and click the **Set Default** button.

The system populates the **`Default`** value in the accumulation table.

NOTE: The system applies the selected tags to the template from which they were selected. If multiple templates have components with the same tag you must select the tag in each applicable template.

Step 10 When the accumulation table contains all the tags you need, click the **OK** button.

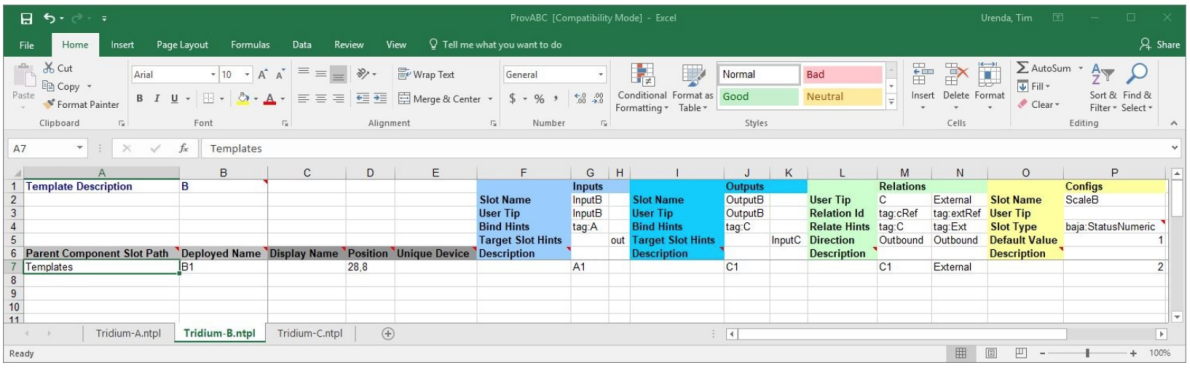
The system creates an Excel spreadsheet in the `templates` folder.

Editing a spreadsheet

Edit the rows in the spreadsheet to provide the unique device information required by each template instance.

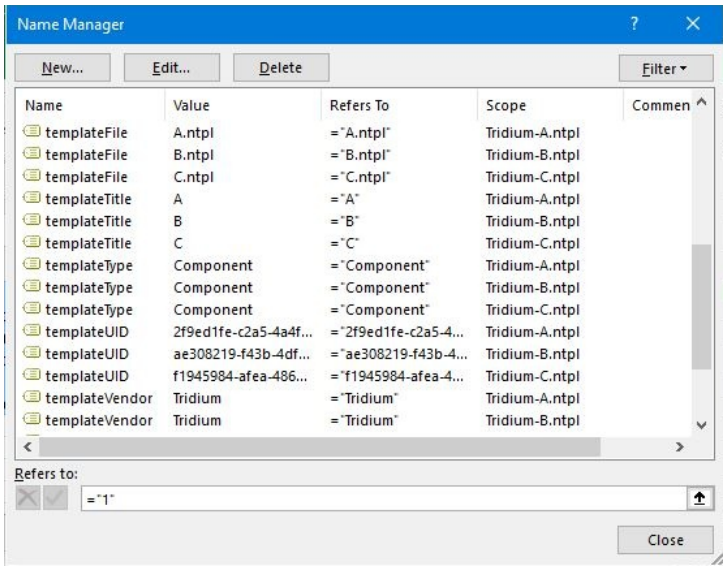
Prerequisites: You previously exported one or more templates and created the template Excel spreadsheet.

Step 1 Open the Excel spreadsheet.



Step 2 To view the formula values, click the **Name Manager** icon in the Formulas ribbon.

The **Name Manager**



The additional information, which comes straight from each template, includes the following:

- Template Filename
- Title
- Vendor
- Version
- UID
- column counts
- export version

CAUTION: Do not change any of the additional information listed above, as the bulk deployment process uses those values to deploy each remote station.

Step 3 Fill the rows in each worksheet tab.

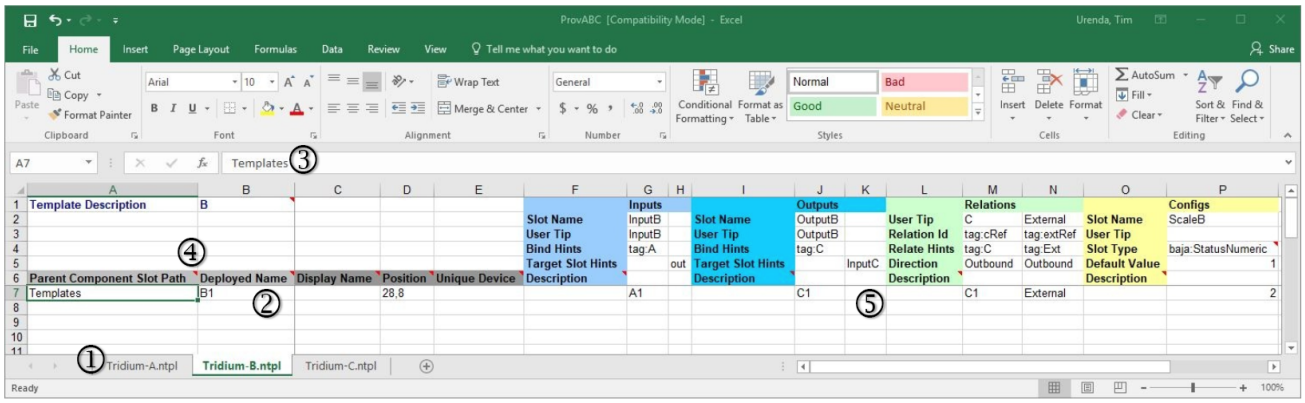
Each row represents a template instance used to configure a device or other station entity.

Parent Component Slot Path and Deployed Name are both required entries. The slot path can contain as many levels as needed. If elements of the slot path do not already exist in the station, the system automatically generates a Folder component for the missing element.

About the exported spreadsheet

A template instance contains the unique data required to configure each device that is managed by a single station. The exported Excel file is populated with metadata and header rows to indicate column positions, which you fill in to correctly configure each template instance.

Figure 1 Example exported Excel file



1	The spreadsheet contains a separate worksheet tab for each selected template. The **`Template Vendor`** and **`Title`** combine to make the worksheet tab name. You can rename and move the worksheet tabs as required. The top six rows of each worksheet contain data required by the bulk deployment process to define the purpose of each column. Do not edit these rows in the spreadsheet. To change their content, go back and edit the template **`Description`** and **`Info`** properties on the **Template Info** tab in Workbench, and export the Excel file again. The order of the worksheet tabs is important. The bulk deployment job works from left to right through each worksheet, adding all template items before creating links and relations. If an individual template requires something else to already exist in the running station, and a template tab installs that something, make sure the required entity is in a worksheet to the left of the worksheet that requires its existence.
2	The left-most columns are information needed to generate a template instance. Metadata and header rows indicate the column positions to configure each template instance. Each of these column headers includes a comment describing the entry for this column. A template instance is defined by filling in a row in the worksheet tab. Additional rows define multiple instances of a template. Values for each column can be required or optional.
3	Additional template information is included as formula values.

4	Comments (indicated by a red triangle in the upper right corner of the cell) provide additional information for specific header cells. Hover the mouse pointer over the cell to show the comments on that cell. If the template has a **`Template Info`** value, it is included as a comment on the template description value cell (B1). If the Excel file is encrypted, a **Privacy Note** label is inserted in cell (A2), with a notice in cell (B2) stating that "This file is encrypted to protect passwords or other private data. Please maintain the encryption and safeguard the contents while editing."
5	The rest of the header columns define inputs, outputs, relations, configuration properties, and (optional) string tags.

NOTE: The string values for most of the generated Excel columns, comments, and metadata are in the template lexicon, those can be localized and internationalized as needed. Refer to the *Niagara Lexicon Guide*.

Template instance data

The template instance data are defined by the left-most columns in the spreadsheet with the gray background. Rows seven through xxxx (however many you need) is where you enter data into the worksheet for each entity, usually a device, but not always. The bulk deployment job processes each row one at a time until, for each row that contains a defined device, it adds a device to the running station.

This table describes the information the bulk deployment job expects you to enter for each device (component).

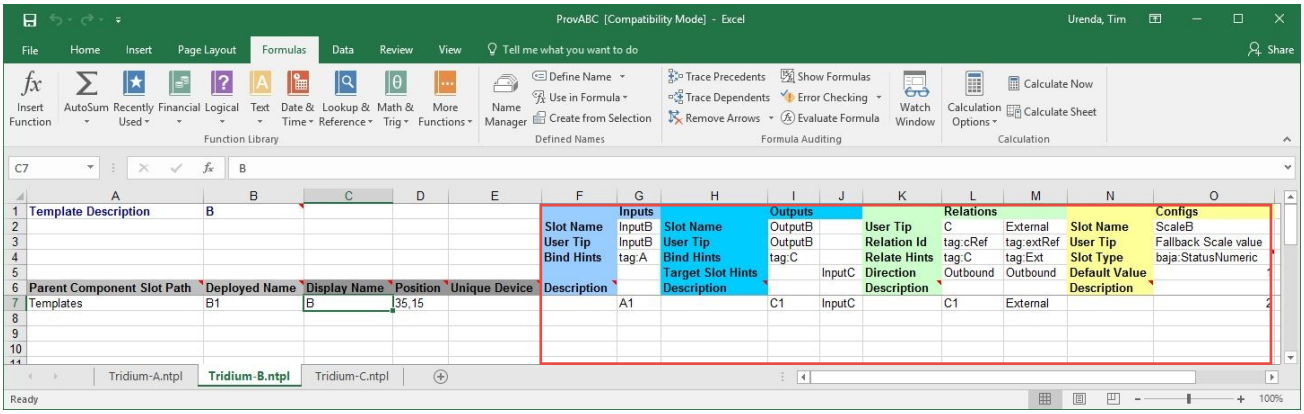
Column	Required	Description
Parent Component Slot Path	Yes	The location in the target station (slot path in Workbench terms) of the device. Include no starting or ending slashes (/) in the entered value, for example: `Folder1/Folder2/Folder3`
Deployed Name	Yes	The name of the device. For example, `AHU_1`
Display Name	No	A more user-friendly name than the deployed name, for example, `Air Handler Unit 1`
Position	No	The deployed component's position on the Wire Sheet. Specifies the position as h,v or h,v,w where h, v, and w are numbers that represents the horizontal, vertical position, and width respectively.
Unique Device	No	The station name or IP address value. This row only applies to a matching station when a provisioning job deploys a bulk template to multiple stations. This value is ignored for template bulk deployment to a single station.
Enable Histories	No	Enables and disables all history extensions in the deployed template whose **`Enabled`** property is not already configured in the template.

Template inputs, outputs, relations, configs, and tags

The right-most header columns of the spreadsheet define inputs, outputs, relations, configs, and tags.

If the source template does not contain inputs, outputs, relations, configuration properties, or tags, columns for those are not included in the exported worksheet tab. For example, if the source template has no defined Relation elements, no Relations columns exist in the spreadsheet.

An initial definition column for each item contains the template details for the item. The details of these definitions are described below. A variable number of columns for each item depend on the source template definition.



The deployment job links inputs and outputs into a slot somewhere in the station.

Inputs

Each template input has a corresponding column for defining the source component for the input. The information in the template is meant to assist you when deciding what the input is used for. The value of the cell is a named component that satisfies the Bind Hints query for that input. The input component must exist on the station for the link to be satisfied. These values are required.

Outputs

Each template output has two corresponding columns for defining the destination component and slot for that output. The value of the cell in the first column is a named component that satisfies the Bind Hints query for the input. The value of the cell in the second column is the target slot name for that named component. The output component and slot must exist on the station for the link to be satisfied.

Relations

Each template relation has a corresponding column for defining the paired component for that relation. The value is a named component that satisfies the Relate Hints query for the relation. The component must exist on the station for the relation to be satisfied.

Configs

Each template configuration property (Config) has a corresponding column for defining the value applied to the Config. Each Slot Type cell for the Config header has a comment that indicates the type of entry to make for the value (integer, number, string, etc.). If you leave the value cell blank for a template instance, the Default Value applies.

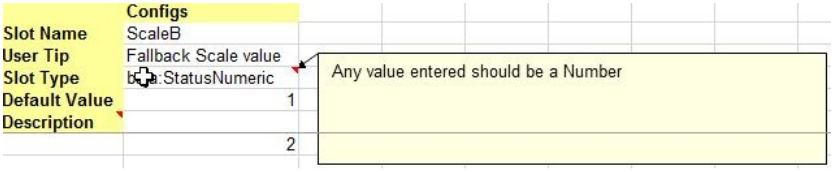


Table 1 Config Data Element

Element	Used In	Description
Slot Name	Inputs, Outputs, Configs	Defines the name of the slot that is added to the template component when deployed. It defines the slot on the template component that inputs link to and outputs link from.
User Tip	All	Provides information about this item to establish the appropriate instance values. This information is meant to assist you when deciding what an input is used for.

Element	Used In	Description
Bind Hints	Inputs, Outputs	Defines the NEQL query string used to derive the candidate components. The NEQL search uses this tag query statement to locate possible components in the running station where the connecting link should be applied.
Target Slot Hints	Inputs, Outputs	Provides a suggestion for source or target slot value of linking component. The value should be the name of a slot that exists on the source or target component.
Relation Id	Relations	Defines the name of the relation, usually chosen from an existing tag dictionary. The deployment job uses this relation definition when it creates the relationship.
Relation Hints	Relations	Defines the NEQL query string used to derive the candidate relation components. The NEQL search uses this tag query statement to locate possible components in the running station where the connecting relation should be applied.
Direction	Relations	Indicates an inbound or outbound relation. This relation direction is based on the device being processed.
Slot Type	Configs	Indicates the data type of the configuration property. The attached comment provides guidance on appropriate values.
Default Value	Configs	Defines the default value to apply if the cell entry if the Config is blank.
Description	All	Provides any additional information about the specific columns. The information in this cell can provide more detailed information as to the purpose of this configuration property. Using this information, a user can determine what values to enter in the worksheet.

Tags

Each requested template string tag has a corresponding column for defining the deployed value for that string tag ID. When deployed, any string tag with that ID inside the template will get the declared value.

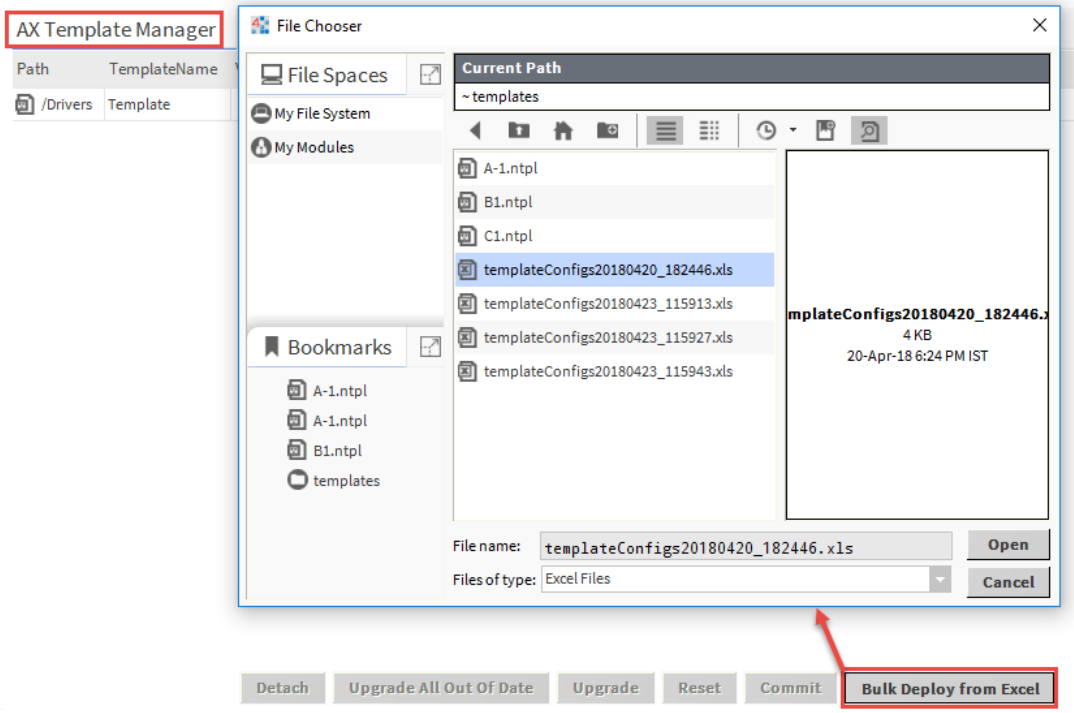
Deploying bulk templates

This topic documents how to import bulk templates to the station database via the **Template Manager** view.

Step 1 Expand **Station→Config→Services** in the Nav tree and double-click the **TemplateService**.

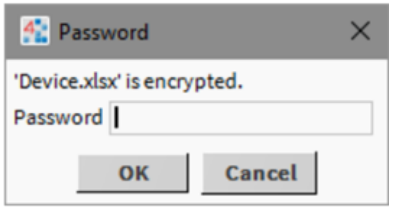
The **Template Manager** view opens.

Step 2 Click the **Bulk Deploy from Excel** button.



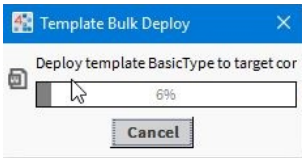
The **File Chooser** window opens.

Step 3 Select an Excel file and click **Open**. If the file is encrypted, a Password window opens. Enter the password to continue.



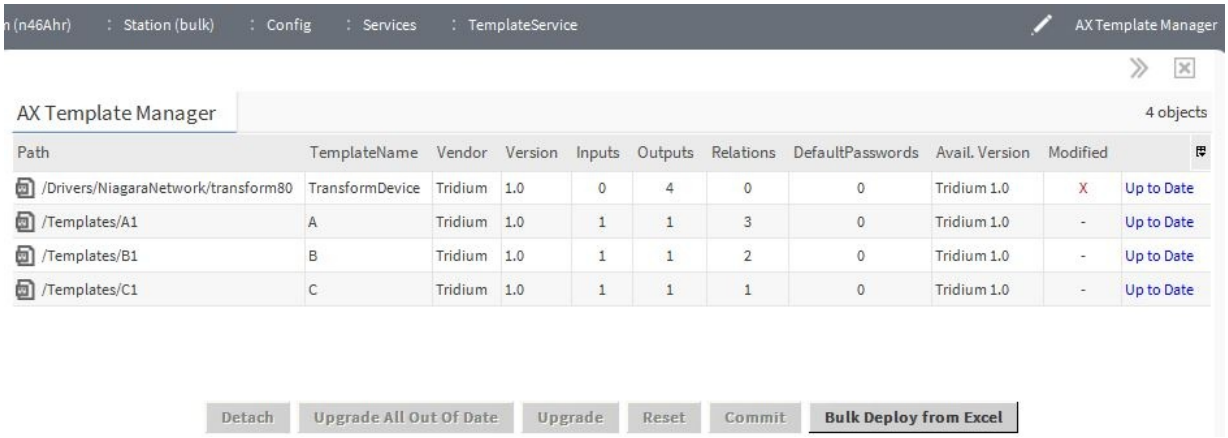
The job deploys the template instances contained in the Excel worksheets to the station. First it generates template components for all template instances. Then it applies the input, output, relation, and configuration elements to the generated template components. This allows links and relations to be established between deployed template instances.

The **Template Bulk Deploy** progress window opens to update the deployment progress.

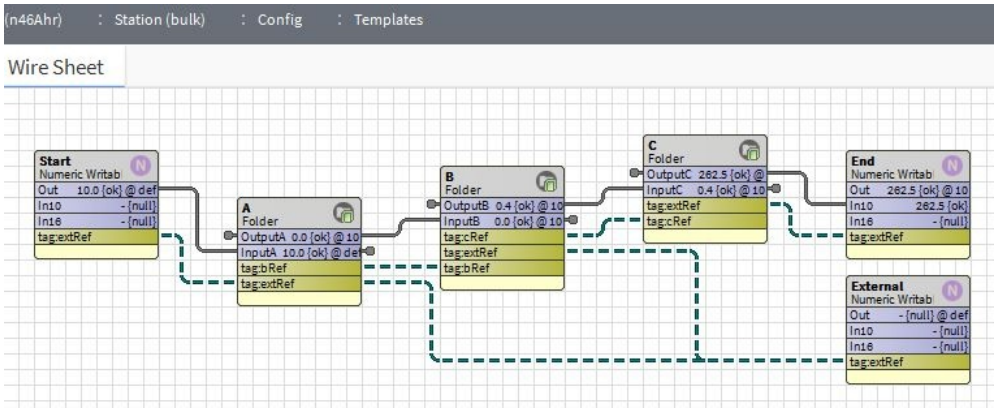


CAUTION: Clicking **Cancel** on the progress window stops the bulk deployment at the point of progress. It does not restore any template component deployments that are already processed.

Once the deployment process is complete, the **Template Manager** view displays the generated template instances.



The Wire Sheet view shows the template components generated, linked and configured.



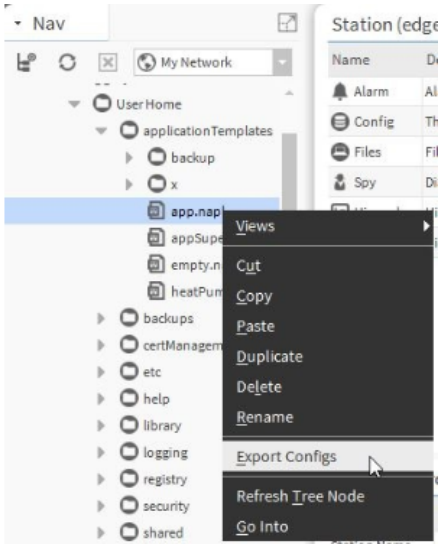
Exporting the application template spreadsheet

You can export the configuration of an application template to an Excel spreadsheet.

Prerequisites: You have a configured application template to export.

Step 1 Navigate to the Workbench user home `applicationTemplates` folder.

Step 2 Right-click on an application template (`.napl`) file and click the **Export Configs** option in the pop-up menu.



The **File Chooser** window opens.

Step 3 Use the **File Chooser** to select a destination and name (extension `.xls`) for the exported file and click **Save**.

The exported results are the same as for exporting component templates to a spreadsheet, except that application templates have no input, output, or relation configurations. For more details, see Exporting the template spreadsheet, page 44.

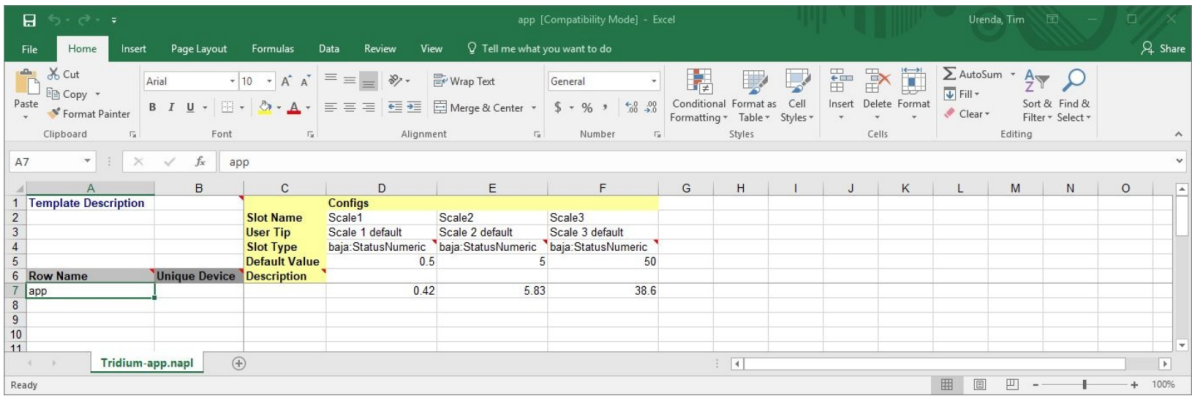
NOTE: Application templates are not available from the **Template Sidebar**.

Editing the application template spreadsheet

You edit the rows in the spreadsheet to provide the unique device information required by the template instance.

Prerequisites: The application template configuration is already exported to an Excel file.

Step 1 Navigate to the spreadsheet and double-click to open it in Excel.



Step 2 Edit any of the configuration values with appropriate entries as needed.

NOTE: One entry is required in the Excel worksheet to generate an instance of the application template during installation.

The exported template configuration results are the same as for component templates, except that application templates have no input, output, or relation configurations. For more details, see Editing a spreadsheet, page 46.

Installing bulk application templates

Niagara 4.8 and later supports installing application templates using bulk deployment. You accomplish this by importing bulk templates to the station database via the **Template Manager** view. The workflow is very similar to deploying component and device templates, with a few key differences that are outlined here.

Prerequisites:

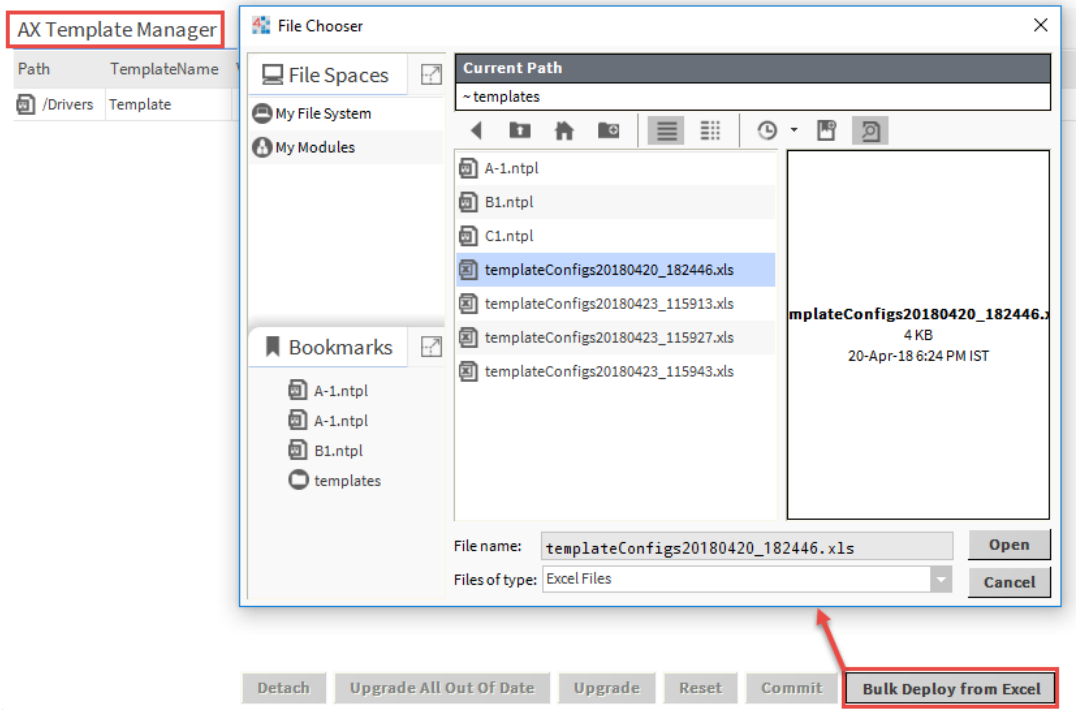
- Application templates will not install to a station where the platform does not have all required modules installed, including any modules containing required PX pages or images.
- Target stations must have the Job and Template services already installed.
- The WB (-wb) runtime profile is required to install templates.

Keep in mind that only one application template can be installed on a station at any one time. Deploying multiple instances of an application template to a station is not meaningful since only the contents of the last application template installed will exist on the station. The real benefit of using the bulk deployment feature is that the template configurations can be stored in an Excel sheet instead of having to enter them by hand during installation.

Step 1 Expand **Station→Config→Services** in the Nav tree and double-click the **TemplateService**.

The **Template Manager** view opens.

Step 2 Click the **Bulk Deploy from Excel** button.



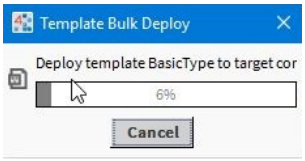
The **File Chooser** window opens.

Step 3 Click an Excel file to import and click **Open**.

The job deploys the template instances contained in the Excel worksheets to the station. First it generates template components for all template instances. Then it applies the input, output,

relation, and configuration elements to the generated template components. This allows links and relations to be established between deployed template instances.

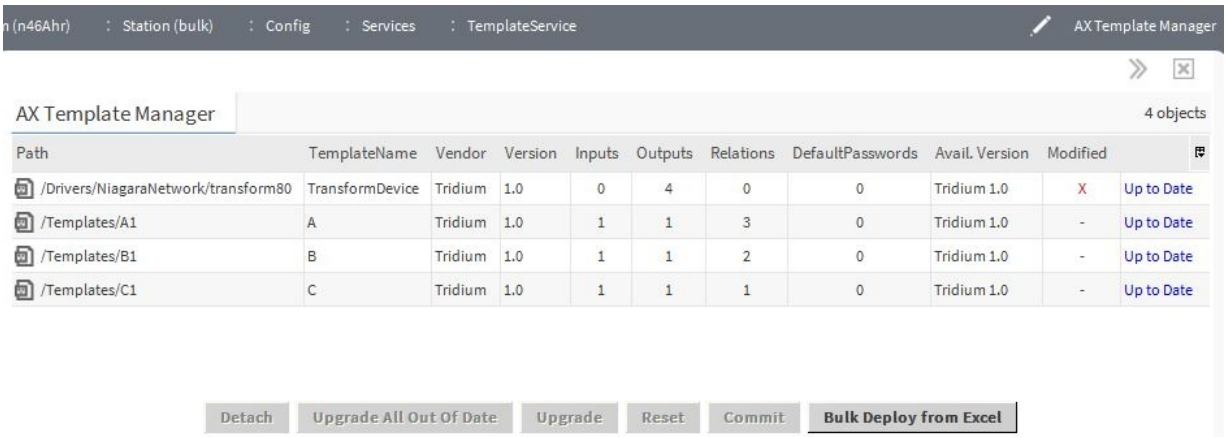
The **Template Bulk Deploy** progress window opens to show the deployment progress.



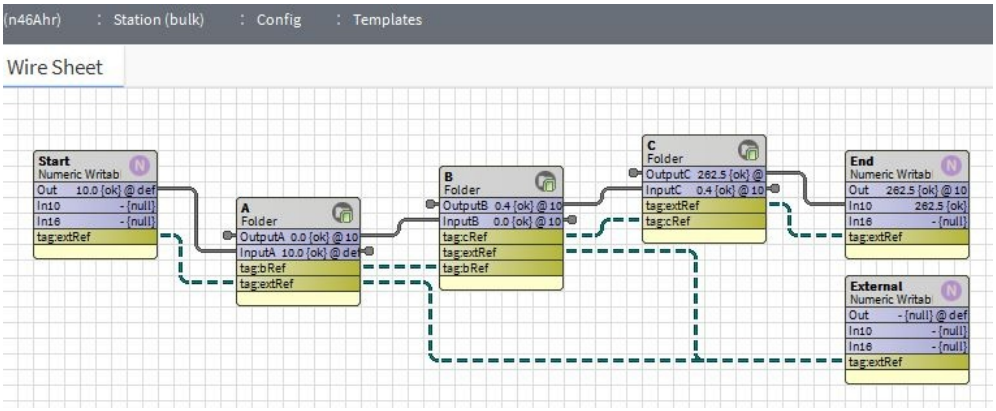
CAUTION: Clicking **Cancel** on the progress window stops the bulk deployment at the point of progress. It does not restore any template component deployments that are already processed.

NOTE: If the Excel file is associated with an application template, the **Install Application Template** wizard will run. It will allow the user to confirm the Excel file selection, choose to make a station backup before installation, and install the application template. It will not prompt for configuration values, but will use the entries from the Excel sheet. It will show installation progress, and notify the user when the installation is complete.

Once the deployment process is complete, the **Template Manager** view displays the generated template instances.



The Wire Sheet view shows the template components generated, linked and configured.

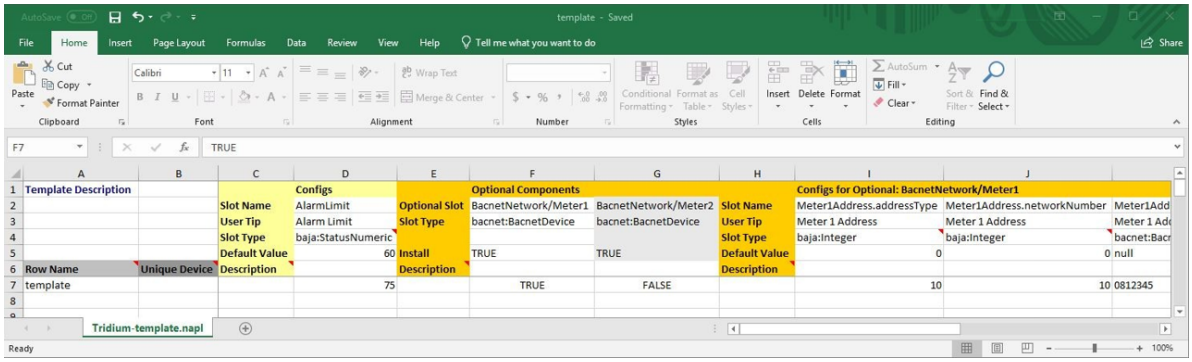


Updating template configuration

In Niagara 4.9 and later, exported spreadsheets used for deploying component templates or installing application templates can also be used to update the configuration of a previously-deployed template.

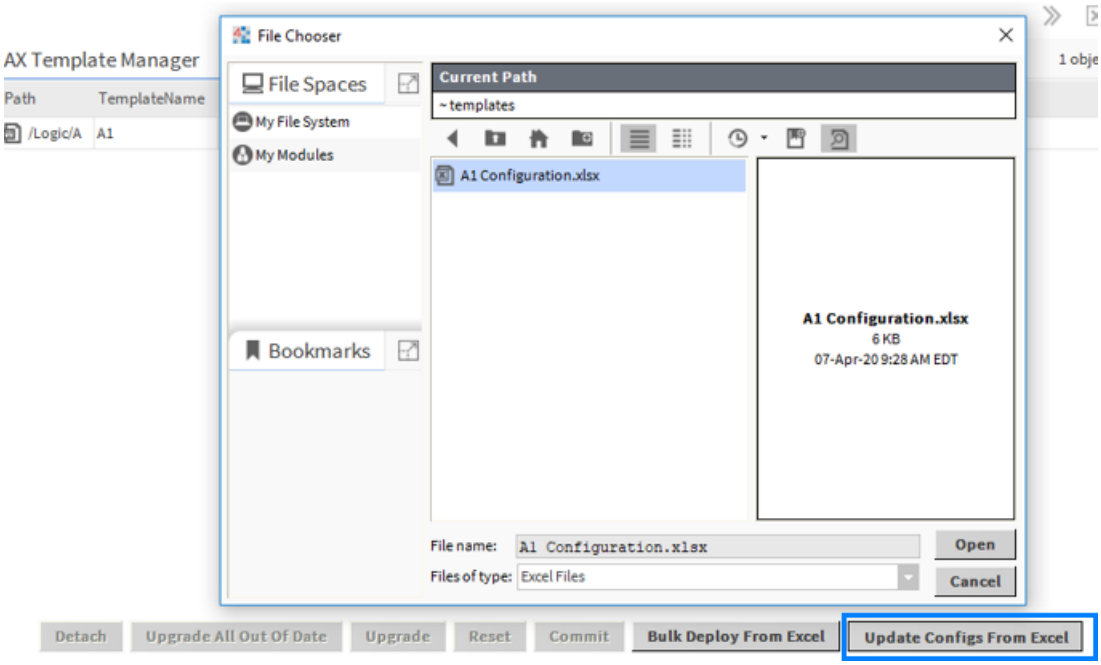
Prerequisites: You already have on hand the exported template spreadsheet.

Step 1 Fill in an exported spreadsheet to describe the template(s) and the revised configuration.



NOTE: To leave any of the configuration as-is, clear the appropriate value cell for the row that represents the deployed template and also clear the default value cell in the header of that same row.

Step 2 In the Template Service's **Template Manager** view, click the **Update Configs From Excel** button to apply changes in the spreadsheet.



Only the template configuration is changed through this process. When updating configuration values, the following logic is used for a given configuration value:

a. If the value cell is not blank, apply the value of that cell.

b. Otherwise, if the default value cell in the spreadsheet header is not blank, apply the value of that cell.

c. Otherwise, leave the current value as is.

NOTE: Note that this process cannot be used to change which optional components are present in an installed application template; for optional components, it can only adjust the configuration of components that were previously installed. To reselect the optional components of an application template, re-install the application template.

NOTE: Updating template and application configurations can also be done through provisioning. For more details, see the latest version of the *Niagara Provisioning Guide*.

Optional components in application templates

In Niagara 4.9 and later, application templates support the declaration and installation of optional components.

Optional components must be declared using the template editor. Only a Device or Device Folder may be declared optional. Optional components can be individually selected for installation during the application template installation workflow.

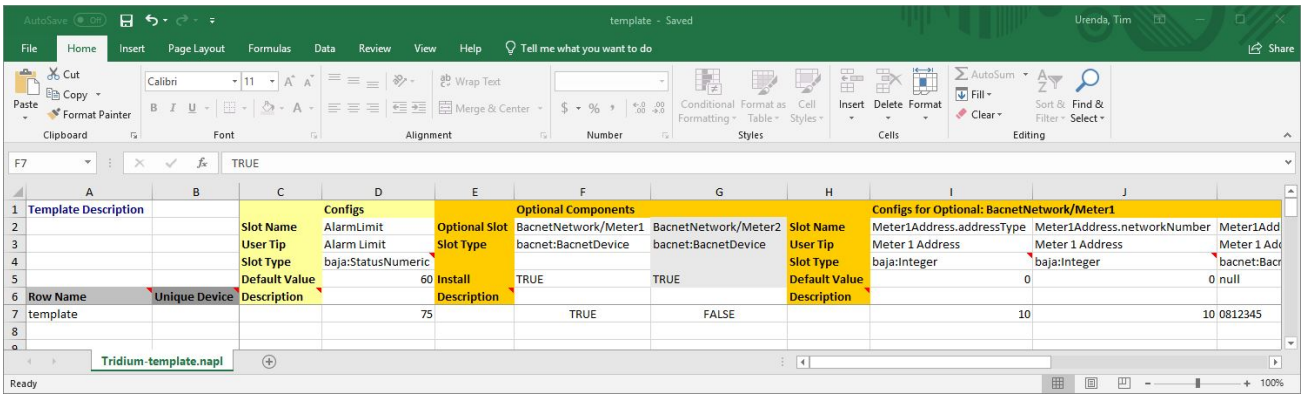
Optional components when installing bulk application templates

Niagara 4.9 and later supports declaration and installation of optional components in application templates using bulk deployment.

There are a few additional steps to the normal installation workflow, as described in the following sections.

Editing the spreadsheet

The exported spreadsheet for application templates containing optional components has some additional features and columns.



	A	B	C	D	E	F	G	H	I	J	
1	Template Description			Configs		Optional Components			Configs for Optional: BacnetNetwork/Meter1		
2			Slot Name	AlarmLimit	Optional Slot	BacnetNetwork/Meter1	BacnetNetwork/Meter2	Slot Name	Meter1Address.addressType	Meter1Address.networkNumber	Meter1Add
3			User Tip	Alarm Limit	Slot Type	bacnet:BacnetDevice	bacnet:BacnetDevice	User Tip	Meter 1 Address	Meter 1 Address	Meter 1 Ad
4			Slot Type	baja:StatusNumeric				Slot Type	baja:Integer	baja:Integer	bacnet:Bacr
5			Default Value	60	Install	TRUE	TRUE	Default Value	0	0	null
6	Row Name	Unique Device	Description		Description			Description			
7	template			75		TRUE	FALSE		10	10	0812345
8											

Each optional component will have a column in the spreadsheet under the **Optional Components** grouping. The value for each entry should be `TRUE` or `FALSE`, indicating whether to install that component or not.

If any optional component has configurations defined on it in the application template, these will have columns grouped by the associated optional component slot. Values for these configurations are defined just like regular configurations. There is alternating shading on each optional component header column that matches the shading on any associated configuration value header columns.

The setup and execution of bulk application template install does not change.

Chapter 6 Components and plugins

Topics covered in this chapter

♦ TemplateService component
♦ TemplateConfig component
♦ Template sidebar
♦ Template Manager view
♦ Template view
♦ Template Options
♦ Example template.manifest.xml file
♦ Template troubleshooting

This section describes the components and plugins for the templates module, as well as some troubleshooting tips.

Components

Components include services, folders and other model building blocks associated with a module. You may drag them to a property or wire sheet from a palette.

The descriptions included in the following topics appear as headings in the online documentation in Workbench. They also appear as context-sensitive help topics when accessed by:

- Right-clicking on the object and selecting **Views→Guide Help**
- Clicking **Help→Guide On Target**

See the following sections on the Template Service component, Template Config component, and Template side bar.

Plugins (Views)

Plugins provide views of components and can be accessed in many ways. For example, double-click a component in the Nav tree to see its default view. In addition, you can right-click on a component and select from its **Views** menu.

For summary documentation on any view in Workbench, select **Help→On View** (**F1**) from the menu or press **F1** while the view is open.

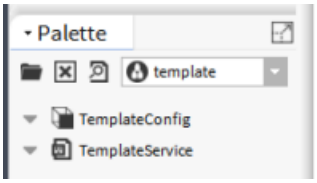
See the following sections on the Template Manager View and the Template view.

TemplateService component

The Template Service provides management support for templates that are deployed in the station. The **Template Manager** is the main view of the **TemplateService**. One of the primary functions of the service is to identify unlinked template inputs, outputs, and component relations and help to resolve them. The Template Service is available in the **template** palette.

NOTE: The Template Service must be installed in the running station in order to take advantage of the templating feature.

Figure 2 Template Service available in the template palette



Type	Value	Description
Status	read-only	Indicates the condition of the component at the last check. `{ok}` indicates that the component is licensed and polling successfully. `{down}` indicates that the last check was unsuccessful, perhaps because of an incorrect property, or possibly loss of network connection. `{disabled}` indicates that the **`Enable`** property is set to `false`. `{fault}` indicates another problem. Refer to **`Fault Cause`** for more information.
Fault Cause	read-only	Indicates the reason why a system object (network, device, component, extension, etc.) is in fault. This property is empty unless a fault exists.
Enabled	`true` or `false`	Activates and deactivates use of the component.

TemplateConfig component

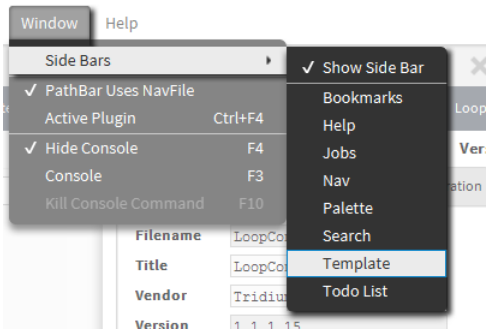
The TemplateConfig component contains a template's configuration properties. Composite links are created from slots in the TemplateConfig component to selected component target slots. TemplateConfig can be accessed in the property sheet view of the station's `config.bog`. TemplateConfig is available in the **`template`** palette.

Property	Value	Description
Template Name	text string	Name to identify the entity
Exposed properties		Slot for each exposed configured property in template
Linkable I/Os		Slot for each composite link in template
PxView(s)		Slot for each PxView contained in template

Template sidebar

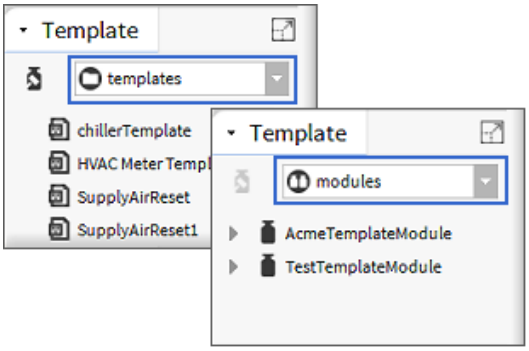
This sidebar provides access to template files located in the Workbench User Home `~templates` folder as well as to templates stored in modules located in the SysHome `!modules` folder.

If not already visible in Workbench, display the sidebar by clicking **Window→sidebars→Template**, as shown here.



In the sidebar, the pull-down list switches the view between the `~templates` and `!modules` folders. When the `!modules` folder is selected, click to expand any module to see the template files contained within.

Figure 3 Template sidebar



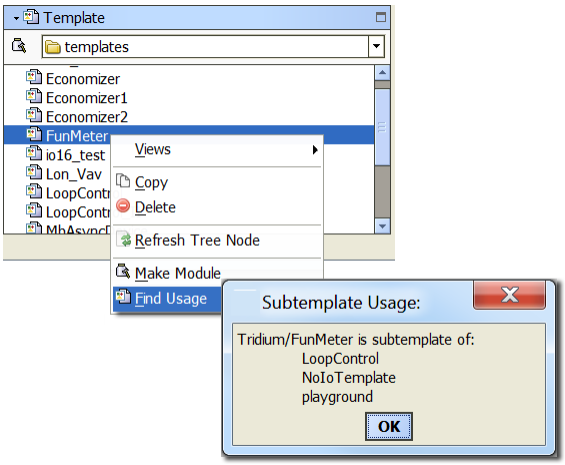
Double-click on a template file to open it in the **Template** view. When you open a file in the templates folder you can proceed to make changes and save the file. Optionally, you can create a new variation of an existing template by clicking **Save As** in the view to save it with a new name.

NOTE: Any template stored in a module is a read-only file which you cannot edit. When you open a template in a module, you will see "ReadOnly" in the top left corner of the **Template** view. To make changes you must first click **Save As** and save the template with a different filename in the Workbench User Home `~templates` folder.

Find Usage option

In Niagara 4.3 and later, the **Template Sidebar** has an added **Find Usage** option on the right-click menu. This menu option invokes the **Subtemplate Usage** window (as shown) which provides a list of parent templates that contain this template. Finding usage is helpful if you have modified a template by adding Inputs, Outputs, or Relations. You should review its usage in the other templates as you will likely need to resolve these new additions within any parent template.

Figure 4 Find Usage menu option invokes Subtemplate Usage window



Template Manager view

The **Template Manager** is the primary view of the Template Service. In Niagara 4.3 and later, added functionality in the manager facilitates upgrading deployed templates by monitoring the version and status of deployed templates, and allowing modifications to the source template. At which point, the upgrade may be applied to all out-of-date deployed templates or only to selected instances. By default, this view

compares the deployed templates against your locally available templates, meaning the templates found in the ~templates folder on your local Workbench PC.

The view displays a table listing all of the deployed templates in the station. In Niagara 4.3 and later, subtemplates are indicated with a path that is indented from the parent template's path. Note that the Path column data displays the deployed component names. The table contains the following information for each deployed template/subtemplate:

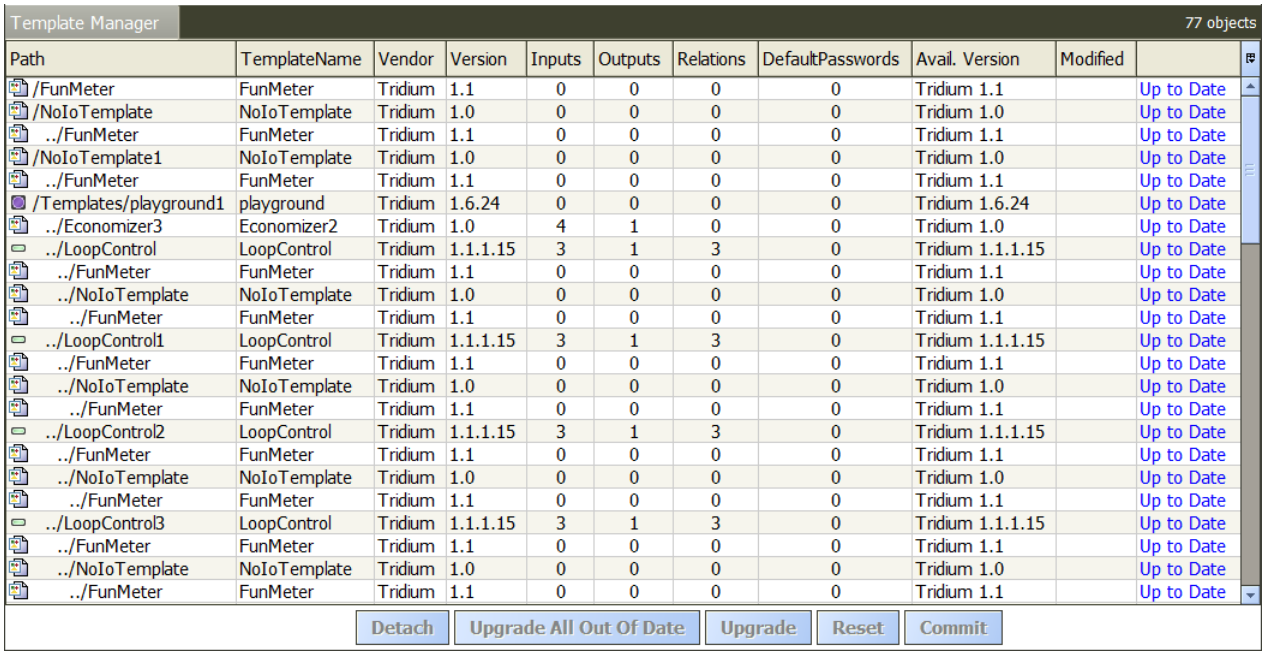
- location, name, vendor, and version
- number of inputs, outputs, and relations, and default passwords,
- available version (of the template on the local Workbench PC)
- modified (detected changes to objects in a deployed template instance)
- status (untitled column, shows the status of the deployed template instance)

NOTE: The Modified column displays a red "X" if something has changed in a deployed template instance. Such a change will be lost if the template is upgraded or redeployed.

Each deployed template listed in the view displays one of the following statuses:

- Up to Date — the version of the deployed template instance is equal to (or higher) than the locally available template version
- Out of Date — the version of the deployed template instance is older than the locally available template version
- Updated —
- Not Available — a deployed template instance is detected however the template file is not in the ~templates folder on the local Workbench PC

Figure 5 Template Manager view compares deployed templates to locally available templates



In the above example, `/FunMeter`, `/NoIoTemplate`; `/NoIoTemplate1`, and `/Templates/playground1` are all "top-level" deployed template instances that are not a deployed subtemplate. All the other rows are the contained subtemplates of the deployed top-level templates.

By examining the Path column you can determine the following:

- The FunMeter template definition does not contain any subtemplates.
- The NoIoTemplate template definition contains a FunMeter subtemplate.
- The LoopControl template definition contains a FunMeter template instance and a NoIoTemplate template instance.
- The playground template definition contains a Economizer3 template instance and 4 LoopControl template instances.

NOTE: If a deployed template has any unbound inputs, outputs, or relations, the view displays those table cells with the Fault background color. Right-clicking one of the Fault rows displays an additional command, **LinkUnboundIO**. Selecting this steps you through the process of linking all of the unlinked inputs, outputs, and relations.

Additional manager functionality

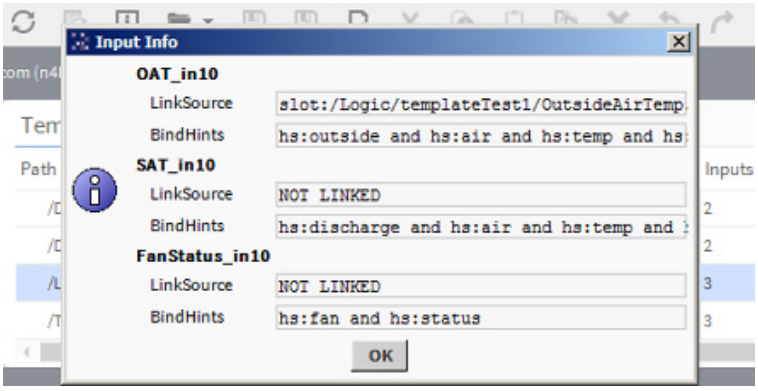
The **Template Manager** provides the following navigational and link resolution functionality:

GoTo navigation

Selecting **GoTo** navigates to the default view of the root component of the deployed template. Invoke the popup by double-clicking on the desired path cell or right-click on the desired path cell and then click on the **GoTo** popup.

Selecting **GoTo Parent** navigates to the default view of the template root component's parent.

Informational popups



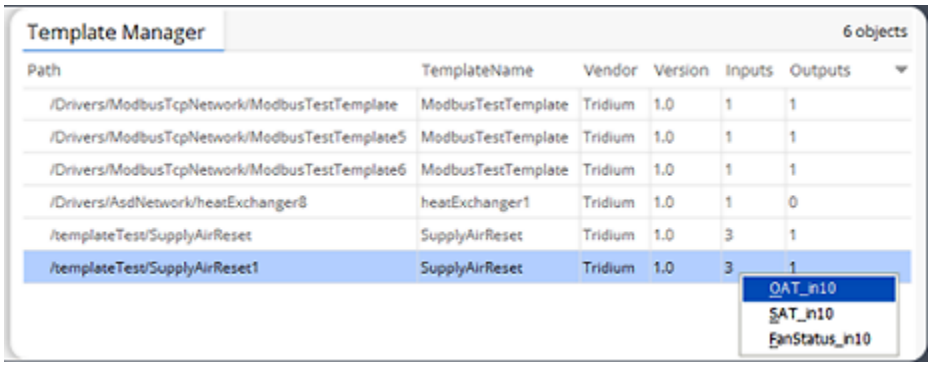
Read-only informational popups can be invoked by clicking table cells in the Inputs, Outputs, or Relations columns in the view. For example, the **Input Info** popup provides you with additional information about the inputs of the deployed template. Invoke the popup by double-clicking an **Inputs** table cell.

In the popup, an unlinked input status is indicated as "NOT LINKED" in the **`LinkSource`** field, as shown.

Double-clicking either an Outputs or Relations table cell invokes the **Outputs Info** and **Relation Info** popups respectively. Referring to the Info popups can be useful when attempting to resolve links and relations. For example, to resolve an unlinked output you can check the bindHints shown in the popup to determine which tags you may need to add to a link target point.

Resolving unbound template I/Os or relations

The **Template Manager** view assists you in linking any unlinked template inputs, outputs, or adding component relations. Right-clicking any cell indicating Fault color selects the row and invokes a popup that lists the unlinked items, shown here.



1. Click an input in the popup to resolve the link.
2. In the **Select Link Input Source** dialog, approve or configure input sources as described here:
   - If the selected input resolves to a single source it is automatically selected for you, simply click **OK** to accept it.
   - If the input resolves to multiple possible sources, you must select one and click **OK**.
   - If the selected input cannot be resolved, a popup appears indicating that fact. Click **OK** to proceed. In this situation, you may need to create an input source point(s) and tag it appropriately so that the input link resolves correctly.

NOTE: The same functionality exists for linking unlinked outputs and relating components.

Batch resolution of unlinked inputs

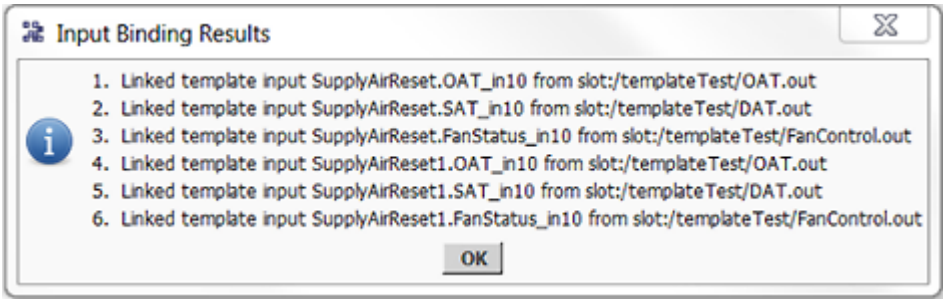
Provides a means of selecting multiple deployed templates and attempting to link all of the selected templates with unlinked inputs or outputs.. To invoke this functionality:

1. Select multiple rows in the **Template Manager** view and then right-click one of the selected rows.

   The LinkUnboundIO popup displays.

2. Click on this to begin the linking process.

   NOTE: During this process the system remembers the choices made by the user for each template type and input. If there are any additional template instances of the same type, the system uses the same choices rather than repeatedly prompting for the same information.
   On completion, a popup dialog displays indicating each linked input and its source.



3. Click **OK** to close the dialog.

Provisioning commands for deployed templates

In Niagara 4.3 and later, the **Template Manager** view supports provisioning of deployed templates. For each deployed template, the **Template Manager** checks for the existence of the template .ntpl file on the client machine. The file's presence and version are then indicated in the "Avail. Version" column in the view. This status modifies the choices of enabled provisioning commands, visible as either dimmed or activated buttons at the bottom of the view..

The following provisioning commands are supported:

- **Detach**

  The Detach function detaches the deployed template from its template definition. Once detached, the deployed component tree can no longer be upgraded. The **Detach** button is enabled when one or more "top level" template rows are selected. Only "top level" templates can be detached. When a "top level" template is detached, all contained subtemplates are also detached. Clicking **Detach** initiates the Detach Process.

  CAUTION: When you Detach a deployed template it is no longer handled as a deployed template, the effect of which is that the template cannot be upgraded in the future.

- **Upgrade**

  If any template is determined to be "Out of Date", the **Upgrade All Out Of Date** button is enabled. Clicking the button flags all out of date templates to be upgraded and then clicking **Commit** initiates the Deploy Process.

  Similarly, if the selected row(s), is "Out of Date" the **Upgrade** button is enabled. Clicking the button flags the selected template(s) to be upgraded. Clicking **Commit** initiates the upgrade process.

- **Downgrade**

  If the selected row(s), is determined to have an "Older version available", the **Downgrade** button is enabled. Clicking the button flags the selected template to be downgraded. Clicking **Commit** initiates the Deploy Process.

- **Redeploy**

  If the selected row(s), is determined to be "Up to Date," the **Redeploy** button is enabled, Clicking the button flags the selected template to be redeployed. Clicking **Commit** initiates the Deploy Process.

Additionally, the Reset and Commit commands are available:

- **Reset**

  Reset is enabled whenever one or more templates are flagged for upgrade, downgrade, or redeploy. Clicking the button removes resets flagged templates to their detected status.

- **Commit**

  The Upgrade, Downgrade, and Redeploy procedures all use the same deployment process to complete provisioning. Clicking **Commit** initiates the Deploy Process.

About the provisioning Deploy process

The Upgrade, Downgrade, and Redeploy provisioning commands all use the same deployment process. This section describes what happens programmatically during this process.

1. Upon clicking **Commit** during the upgrade, downgrade, or redeploy provisioning procedures, a **Commit Template** confirmation window alerts the user that this process removes the existing template and installs a different version, and prompts the user to click **Yes** to confirm that he/she wishes to continue.
2. If the target is an online station (this is handled by an Upgrade Job in the station).
   a. For each template that has been selected to be upgraded, downgraded, or redeployed the following steps occur:

i. The template .ntpl file is copied to the target station's file system at: `^template/"+vendor`. If it already exists (validated with crc check) the file is not copied.

ii. Adds a reference to the template root to be updated to a BVector that will be an argument to the Upgrade Job.

b. Submits an Upgrade Job with a list of template roots to be upgraded.

c. The Upgrade Job runs in the station.

i. For each template identified to be upgraded the following occurs:

1) Get a reference to the .ntpl pushed to the station's file system (which was pushed by client prior to launching this job)

2) Get a reference to the deployed template root component's parent.

3) Save the current state of the deployed template:

- Save the location of the deployed template root component (BWsAnnotation)
- Save persistent properties of the template root component.
- Save the location and values of any BPassword slots.
- Save the current value of template configuration properties
- Save any input links to the template root component.
- Save any output knobs from the template root component
- Save any input relations to the template root component
- Save any output relations from the template root component
- Save the location and value of any tags in the template component tree that aren't defined in the template definition and namespace is not "ntpl".
- Save the location and current value of any slots defined as an editable slot in a PxView.
- Save any external links and relations to or from internal template components.
- Save current state of any BHistoryExt's

4) Remove the existing template root component from its parent.

5) Using the .ntpl reference deploy the template to the template's parent.

6) Restore the values saved in the previous Save step.

In general, the restoration of saved property values is done by property name and value. If the template property has been renamed and it is one of the saved properties, its value will not be restored.

- Restore the location of the deployed template root component (BWsAnnotation)
- Restore persistent properties of the template root component. If properties have been removed from the template they will not b restored.
- Restore the location and values of any BPassword slots. If BPassword properties have been removed from the template they will not be restored.
- Restore the current value of template configuration properties If a template configuration property has been removed it will not be restored.
- Restore any input links to the template root component.
- Restore any output knobs from the template root component
- Restore any input relations to the template root component
- Restore any output relations from the template root component

- Restore any save component tags.
- Restore the saved values of any slots defined as an editable slot in a PxView.
- Restore the saved external links and relations.
- Restore the saved state of BHistoryExt's.

3. The online station deployment is complete.

About the provisioning Detach process

The Detach process may be used when you need to make modifications to a deployed template. This section describes what happens programmatically during this process.

CAUTION: When you Detach a deployed template it is no longer handled as a deployed template. This means that the template cannot be upgraded in the future.

Note that any links and relations will be maintained.

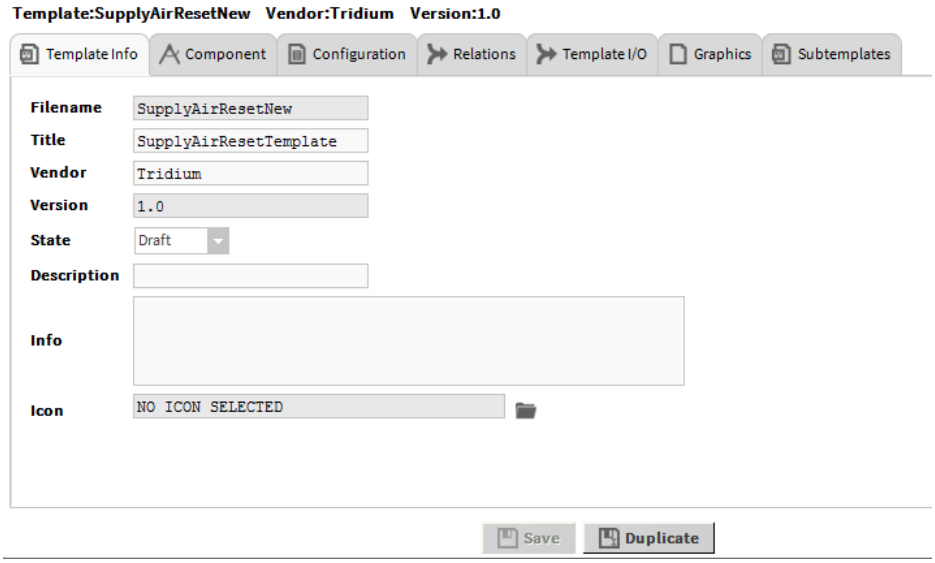
1. Upon clicking **Detach** the **Detach** confirmation window alerts the user that this will detach the selected deployed templates and that detached templates cannot be upgraded. The window prompts the user to confirm if he/she wishes to continue by clicking a checkbox and the **Yes** button.
2. Removes the TemplateConfig component from the deployed template root component.
3. Removes the "ntpl" tags from the deployed template root component.
4. If "Remove Composite Slots" is selected,
   a. for each root component composite link (template input)
      i. Creates a new link to eliminate the composite slot and removes the composite slot and adds a new link to the internal target.
   b. for each root component composite knob (template output)
      i. Creates a new link to eliminate the composite slot and removes the composite slot and adds a new link to the external target.
5. Completes the detach process.

Template view

The **Template View** is the primary view for creating and editing templates. Invoked when you right-click a component and select **Make template** or when you double-click on an existing template either in the **Template** side bar or in the My File System. The tabs present in the view allow you to configure template *.ntpl files. These same tabs are used during the template creation process or to make template modifications.

Template Info tab

This tab allows the designer to enter the basic information that uses to identify and monitor the status of the template.



You have the option to either **Save** or **Duplicate** the template. When duplicating an existing template you are prompted to save the template with a new name. The new *.ntpl file is saved to the ~templates directory.

In addition to the properties listed below, any of these notices may display on the **Template Info** tab:

- If the template component tree contains links whose source or target components are not contained within the same component tree, the editor automatically generates template inputs and outputs to match each. Further, the template editor automatically sets the **`bindHints`** for each input or output from the marker tags that are present on the external components. A notice alerts you to review the added links in the **Template I/O** tab.
- If the template root component is a Device component, a notice alerts you that:

  1. This is a new device template.
  2. Review the **Component** tab.
  3. Set the device address property to its default value. This is important because the device address property and its default value are device specific.

  This check helps to ensure that when the device template is deployed using the **Device Manager** view that the "Match" operation is successful.
- If the template component tree contains a **`Password`** property, a notice alerts you that the template has internal passwords that may need to be set during the deployment. For security purposes, when passwords are copied from a template to a station, the password value may be set to `Password.DEFAULT`. In this case, during the deployment the user must assign a valid value to the **`Password`** property.

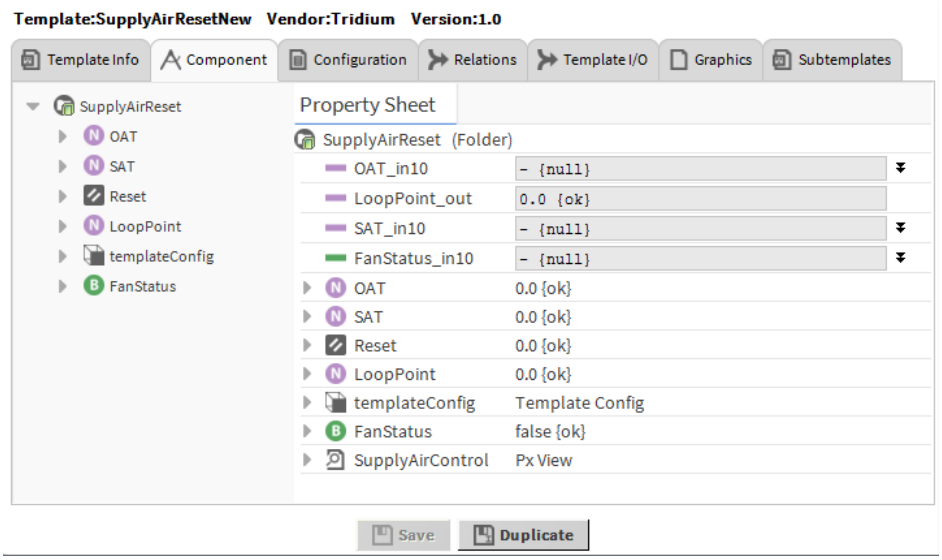
Property	Value	Description
Filename	text	Displays the filename which is the same as the root component. This can be modified during the creation process. Once the template is saved, the filename becomes read-only.
Title	text	Provides the title of the template.
Vendor	text	Provides the name of the device vendor who is responsible for the design and creation of the template.
Version	read-only (number) (defaults to 1.0)	Displays the template version number.

Property	Value	Description
		Upon changing the template **`State`** property from `Draft` to `Ready` the field becomes editable and you are prompted to increment the version. The value entered must be greater than the previous number. Otherwise, version number is user-defined. For example, you might increment the number from 1.0 to 1.1, or to 2.0.1, etc.
State	drop-down list (defaults to `Draft`)	Displays the template state. This is true whether making a new template or editing an existing one. Setting the **`State`** to `Ready` makes the template eligible for deployment to a station. When set to `Ready`, you are instructed to increase the version number.
Description	text	Provides the short description (up to 25 characters) to distinguish similar templates. This name is displayed in the **Device Manager** template side pane.
Info	text	Gives the additional explanation to describe the template. Multiple lines are allowed. This text is visible in the template. manifest.xml file.
Icon	path	Allows you to associate the template with a .png file. Typically an icon file. This icon displays in the **Device Manager** template side pane.

Component tab

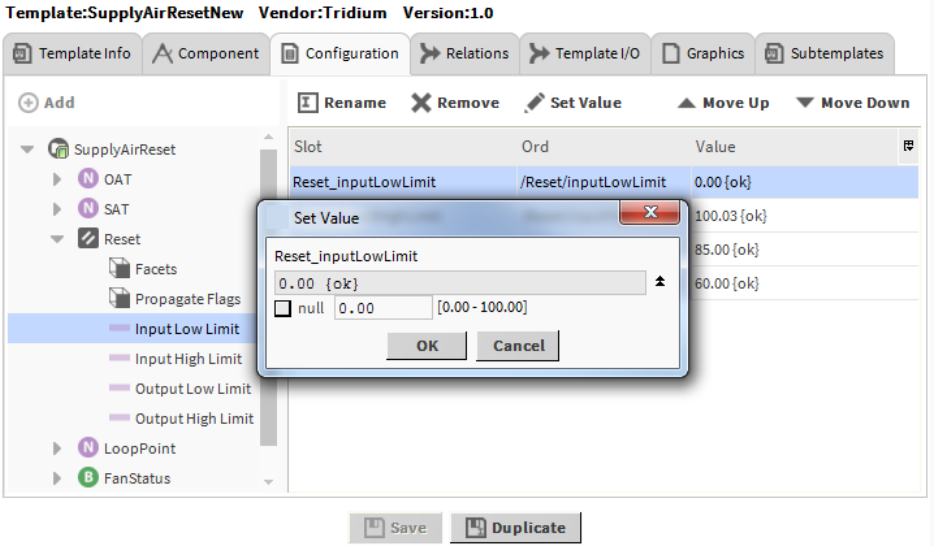
This tab shows the `config.bog` settings for the base component and its descendents.

The tab contains two panes. The left pane shows a tree view of the components that is similar to the Nav sidebar. The right pane shows a Property sheet view of the component.



Configuration tab

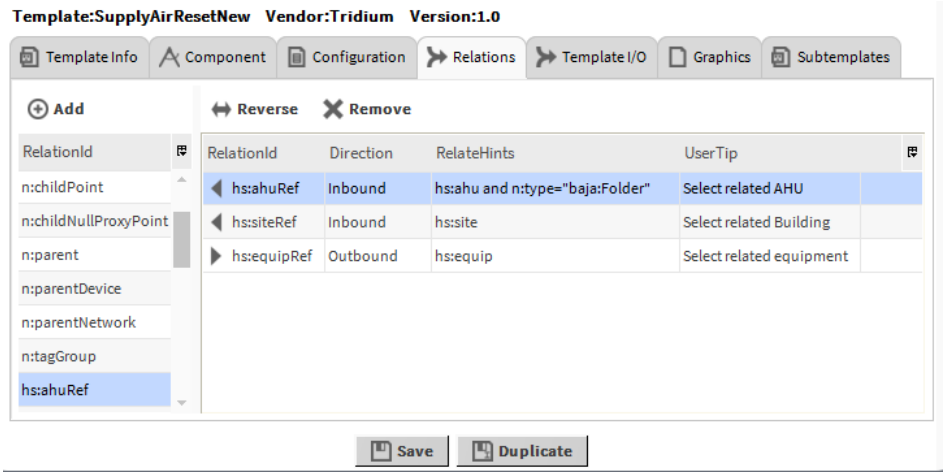
This tab allows the designer to configure the properties that are presented to the user. During the deployment, an edit window prompts the user to set values for specific properties.



This tab contains two panes. The left pane shows the template's component tree. The right pane shows the current set of configuration entries. The configuration entries result in properties of the same type in the **`templateConfig`** object on the base component.

Relations tab

This tab allows you to define relations between the root component of this template and other components in the station in which the template is deployed.



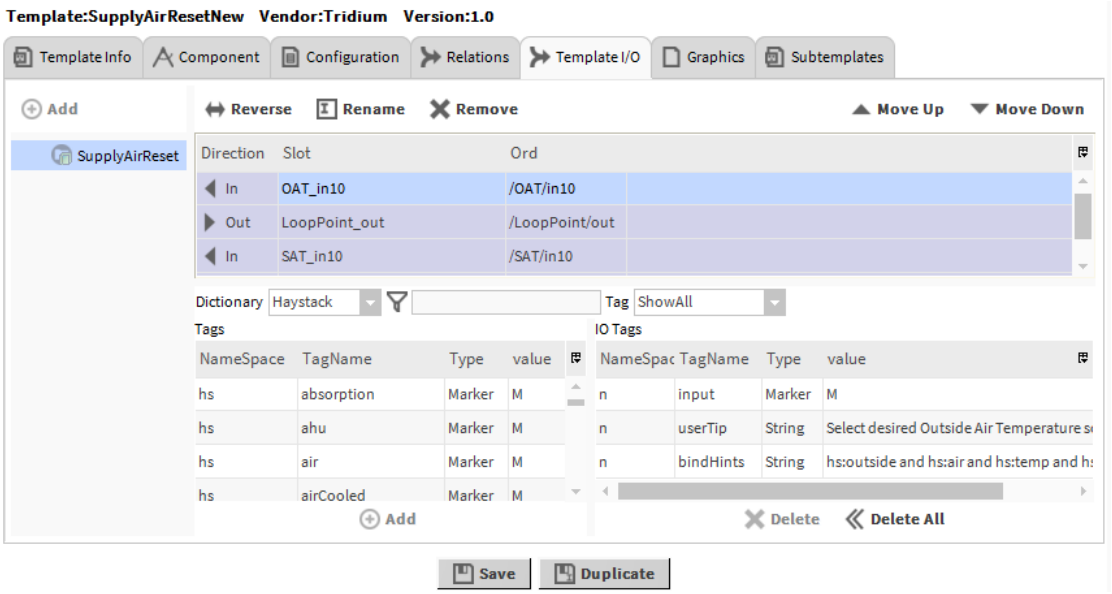
The tab contains two panes. The left pane provides a list of all Relations Id's found in the installed tag dictionaries. The right pane displays a table listing any relations that have been added to the template. Use the buttons above the panes to Add , Reverse , and Remove relations.

You can add any relations which are defined in the tag dictionaries installed on your system by selecting from the left pane and clicking Add . Optionally, you can add Ad Hoc relations by clicking Add without first making a selection, and defining the RelationId in the **Add** window. By default, the Relation Direction of an added relation is "inbound". Clicking Reverse changes the relation direction of the selected Relation. Select a relation row in the right pane and double-click the **RelateHints** value cell to enter a NEQL search predicate. This NEQL search is used during template deployment to search for and suggest potential related components.

Template I/O tab

This tab allows the designer to manage linkable control points in the component tree of a device template.

You can select and expose control inputs that are required by the template and control outputs provided by the template. Also, you can add and remove tags on the control points. The primary purpose for tagging the inputs and outputs is to assist the installation tool during deployment in the resolving links.



NOTE: This tab is intentionally omitted from the station templates since you cannot configure I/Os for a station template.

The **Template I/O** tab contains three panes:

- Left side pane—shows the root component.
- The upper right pane—shows the Slot sheet listing the template's current control inputs and outputs.
- The lower pane—shows on the left side allows you to select additional tags from available dictionaries, and on the right side allows application of tags to selected I/O points.

In the nav pane you can click to expand control inputs, select one and click **Add** to add it to the upper right pane, exposing that input for linking or you can simply double-click on it in the left pane to have the same effect.

Selecting a control input in the upper right pane displays (in the lower right pane) any existing tags on that input. If desired, use delete options in the right-side lower pane to remove tags. Use the left-side lower pane to select additional tags from available dictionaries to add to the input that is currently selected in the upper right pane.

Exposing control outputs is performed in the same manner as inputs.

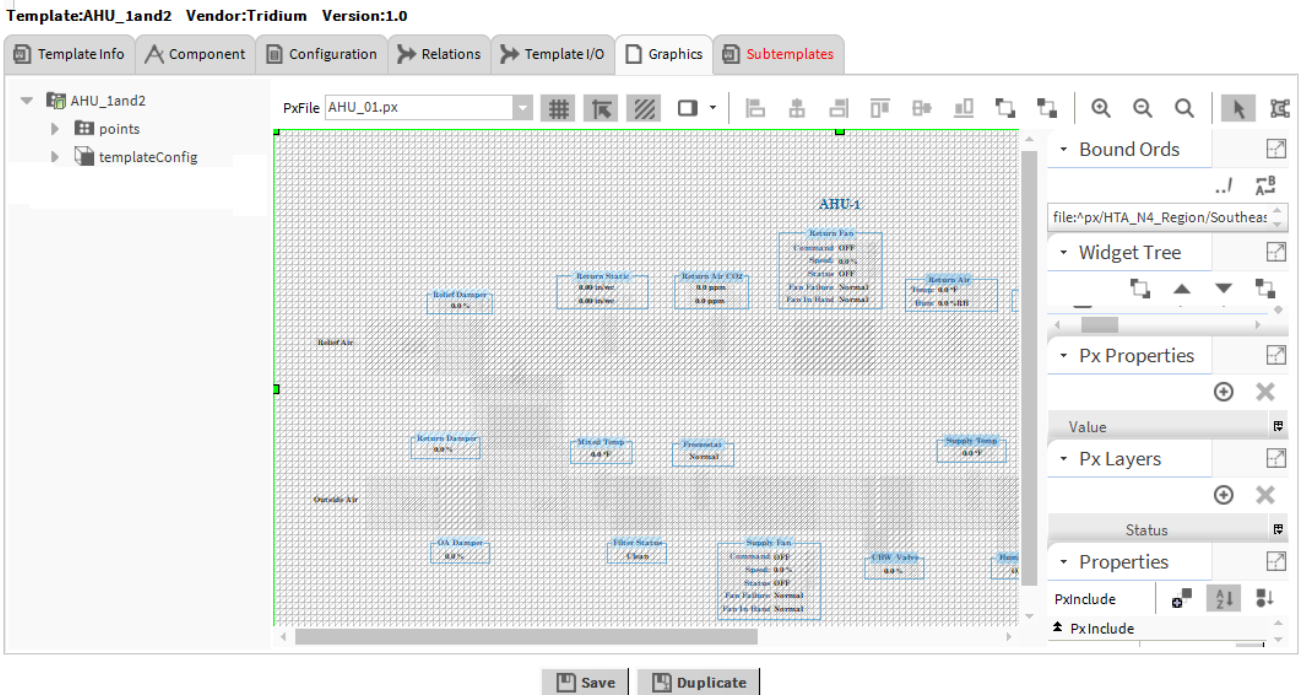
Graphics tab

The **Graphics** tab is a wrapper view of the **PxEditor** view.

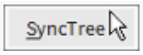
The best practice in creating and editing graphics is to fully engineer them in a station before making the template. This is especially true if image files are used as they will likely not be visible in the template graphic editor. Images from `.jar` files will be visible in the template editor. Use the template graphic editor to make minor changes to a graphic.

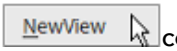
In the view, the left pane is the nav pane and the right pane is the pxEditor pane. If there are multiple Px files contained in the template the **PxFile** drop-down list allows you to choose the PxFile to edit. Selecting a

PxFile loads the file into the editor and selects the component(s) in the nav pane that have a PxView component referencing the selected PxFile.

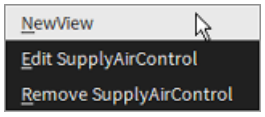


Nav tree controls

- Right-clicking in the nav pane white space provides a 

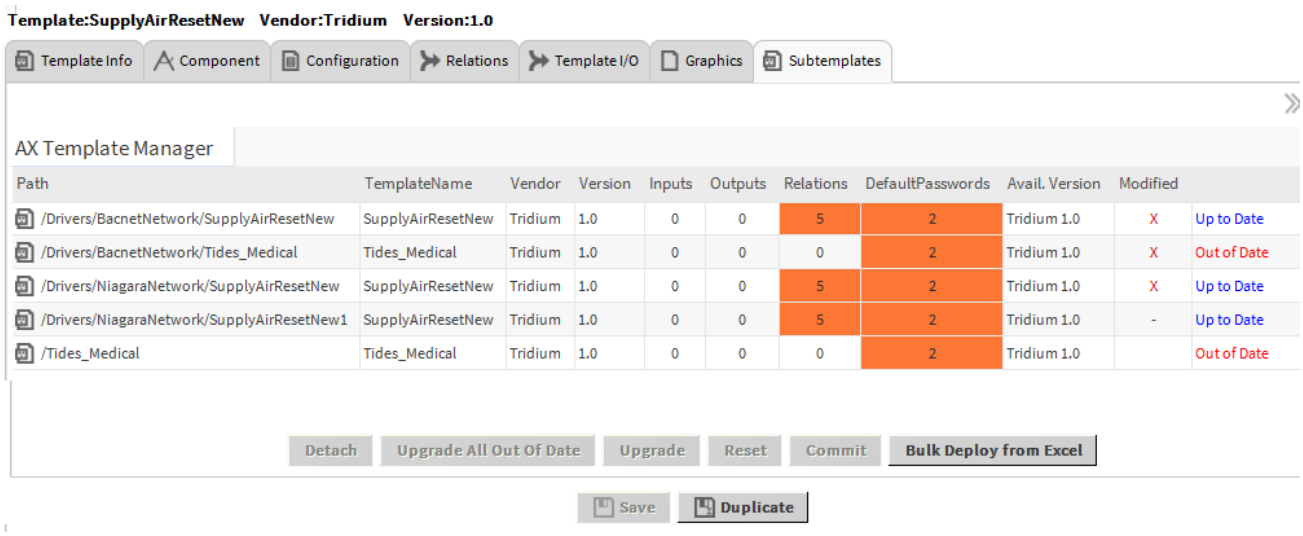
command popup which selects components in the tree that have a view using the currently selected pxFile.
- Right-clicking on a nav component that does not have a view defined provides a 

command popup. Selecting NewView allows you to apply an existing pxView to this component or to create a new pxView.
- Right-clicking on a nav component that already has a view defined displays a popup menu of available views. From this menu, you can create a **NewView**, **Edit** a view or **Remove** a view. If the nav component has multiple views defined, there are multiple Edit and Remove commands listed, one for each view.



Subtemplates tab

The **Subtemplates** tab displays a list of any subtemplates contained by the parent template. This tab uses the **Template Manager** view to manage the subtemplates deployed in a template.



When the **Template View** loads a template it checks for subtemplate instances and determines if there are newer versions available. Any "Out of Date" subtemplates are highlighted the **Subtemplates** tab. Any updated subtemplates are flagged as "Up to Date".

Click on **Save** to save the updated subtemplates.

NOTE: When a template is deployed it always uses the latest version of any contained subtemplates found on the client Workbench even if the parent template has not upgraded its subtemplates. If a template has been modified to add additional Inputs, Outputs, Relations, or Configuration properties, it would be important to review any templates that use the modified template to resolve the new I/Os within the parent template.

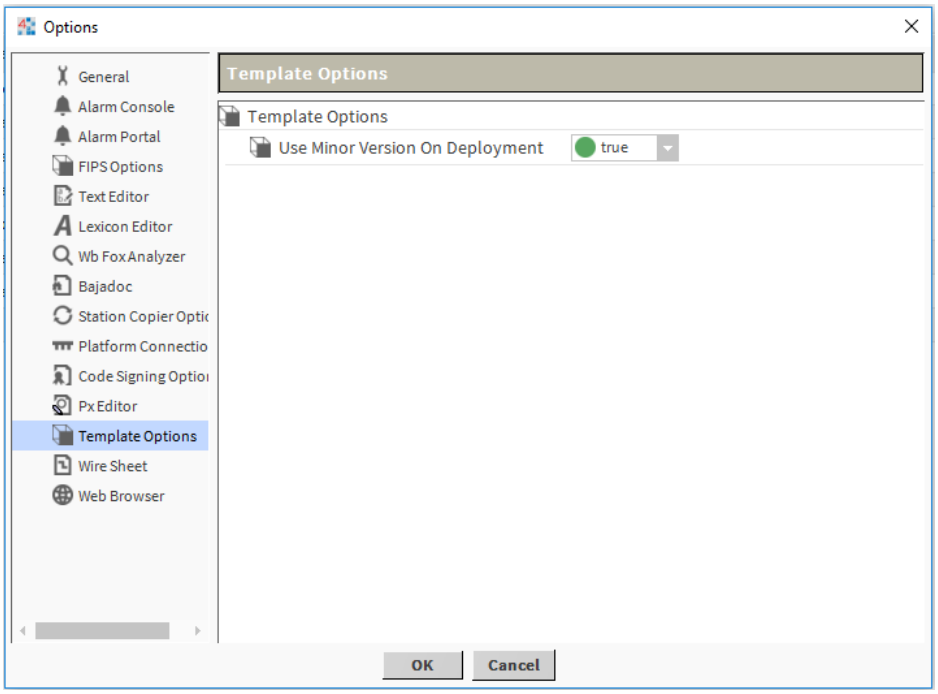
Template Options

This option is used when template files are generated. It determines whether the template should include minor versions of Niagara modules after identifying the component and Px file dependencies.

If **`Use Minor Version On Deployment`** is set to `true`, a template created in Niagara 4.8 have dependencies on minor version of Niagara 4.8 modules.

If **`Use Minor Version On Deployment`** is set to `false`, the template states dependencies on the major version of Niagara 4 modules. This allows template that is created in later versions to be deployed to stations running previous minor versions.

Figure 6 Workbench Template Options



Example template.manifest.xml file

The example template.manifest.xml file shown here reflects the changes that are present in Niagara 4.3 and later.

Initially, templates were identified by filename. In Niagara 4.3 and later, templates are identified by a unique number, the template id number. Assigned at creation, the template id number is never changed.

The template manifest contains a `<subtemplates>` section that is populated with an entry for each subtemplate instance contained in this template. The example shows that the file contains an entry for the Funmeter subtemplate instance.

The template manifest `<dependencies>` section is populated with modules that this template is dependent upon. This should include both control logic, i.e. component tree, dependencies as well as any Px graphic dependencies.

Additionally, the “signature" attribute in the `template.manifest.xml` file helps to determine if a template has been modified.

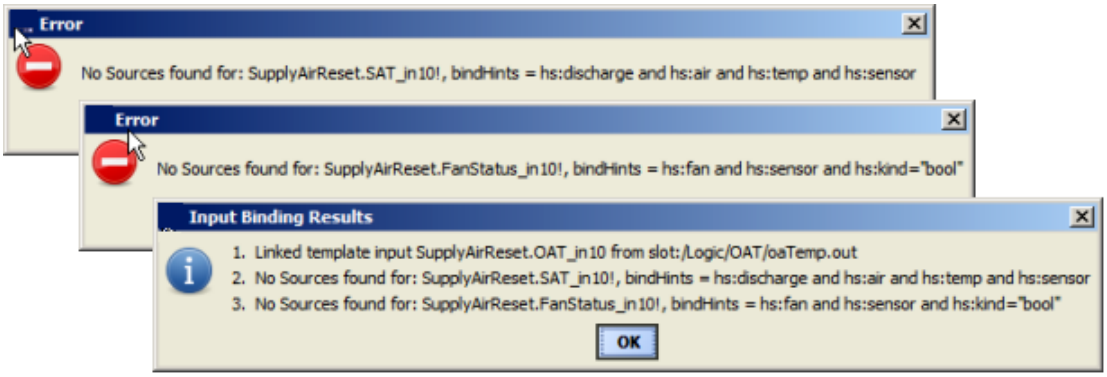
Figure 7 Example template.manifest.xml file

```
<?xml version="1.0" encoding="UTF-8"?>
<template id="ff99a265-10ae-4b3f-b957-91cdc479c9a2" version="1.0" vendor="Tridium" description=""
 state="10" buildVersion="4.3.40.1" signature="860b01d3">

  <info i=""/>

  <settings>
    <p n="version" req="true" typ="str"/>
    <p n="versionDate" req="true" typ="str"/>
    <p n="deployed" req="true" typ="bool"/>
    <p n="pxEditBindings" req="true" typ="str"/>
  </settings>

  <links>
  </links>

  <bindings>
  </bindings>

  <resources>
  </resources>

  <subtemplates>
    <s n="FunMeter" vendor="Tridium" version="1.1" locationOrd="slot:/Economizer/FunMeter"
     ntplFileOrd="file:~templates/FunMeter.ntpl"/>
  </subtemplates>

  <tags>
  </tags>

  <dependencies>
    <dependency name="control-rt" vendorVersion="4.3.39.1" rel="minimum" vendor="Tridium"/>
    <dependency name="platform-rt" vendorVersion="4.3.40.1" rel="minimum" vendor="Tridium"/>
    <dependency name="baja" vendorVersion="4.3.40.1" rel="minimum" vendor="Tridium"/>
    <dependency name="template-rt" vendorVersion="4.3.40.1" rel="minimum" vendor="Tridium"/>
    <dependency name="kitControl-rt" vendorVersion="4.3.40.1" rel="minimum" vendor="Tridium"/>
  </dependencies>

</template>
```

Template troubleshooting

The section provides examples of issues you may encounter using templates and recommended steps to resolve the issues.

While deploying a template you got the following errors but the template deployed anyway:



The template is working correctly. The errors messages alert you to the fact that the deployed template is seeking inputs that have the referenced tags applied however, those inputs are not found. If the necessary input source points are not contained in the deployed template, you may need to add them, apply the specified tags, and resolve the links. The input source points can be located anywhere in the station.

NOTE: Note the same functionality exists for undetected output link targets and component relations.

The Input Binding Results dialog in the above image shows that the template is unable to locate input sources for Reset_SupplyAirTemp and Reset_FanStatus. To resolve those links do the following:

1. In the wiresheet view of the parent of the deployed object, add the necessary input source points, for the first one right-click and add a **Numeric writable point** named SAT.
2. Right-click the added numeric writable point and click **Edit tags**.
3. Select the Haystack tag dictionary and in the search field enter the letter "`d`" for (discharge air temp).
4. In the resulting tag list, scroll to **Tag Groups** and select the **dischargeAirTempSensor** tag group.

   TIP: Using Tag Groups lets you add multiple tags at once. For example, the **dischargeAirTempSensor** tag group applies the following tags: discharge, air, temp, and sensor.
5. Repeat the steps 1–5, this time adding a **Boolean writable point** named FanStatus. In the **Edit Tags** dialog, search the Haystack tag dictionary for tags starting with the letter "`f`" and from the **Tag Groups** list select the **fanSensor** tag.
6. Click **Save.**
7. Next, navigate to the station's **Template Service** and double-click to open the **Template Manager** view. The unlinked inputs for this template have the Fault background color. Right-click there to resolve the unlinked inputs and click **OK**. Repeat as needed to resolve all unlinked inputs.

The template is successfully deployed once all input links are resolved.

You deployed a template to a device and it did not match the device to the template. So it does not work.

There is a mismatch between the device template and the device you are using. You may have deployed a template that was designed for a different type of device.

If it is a programmable device, set up with a different collection of points, that doesn't match the template type, it will not work.

Deploying a template using the Device Manager method is useful since it filters and matches the device to appropriate device template(s). To do this, go to the **Device Manager** view and perform **Learn** (identifies devices connected to the network), and **Match** (matches any unbound templated devices).

Glossary

namespace	A container for a set of names in a naming system. A tag dictionary is a namespace.
NEQL	Niagara Entity Query Language provides a simple mechanism for querying objects with tags. Whereas BQL supports the tree semantics and pathing of Workbench component space (for example parent.parent) and BFormat operations, NEQL queries only for tags using the Niagara 4 tagable and entity APIs.
niagara_user_home	The location(s) under which all configurable data resides. Included are stations, templates, tag dictionaries, registry, logs, and other data. Unique to Niagara 4, the Windows host usually has at least two niagara_user_ homes: • Workbench User Home, which you see in Workbench as "**User Home**". • "Daemon" user home, which you access using a local platform connection. For more details, see "Niagara 4 directory (homes) architecture" in the *Niagara Platform Guide*.
relation	A piece of semantic information (metadata) that indicates how components are related to each other. Relations provide metadata used primarily in building hierarchies for logical views of your system based on relationships between components. You add relations between components for purposes of building hierarchies. Optionally, adding one or more tags to a relation provides additional metadata which allows for more specific filtering when building hierarchies.
station template	A station template is a specialization which contains a complete set of configured objects, everything required for the initial starting point for a new station. The **New Station** wizard in **Tools** utilizes default station templates (NewControllerStation.nptl and NewSupervisorStation.nptl) as well as any user-defined station templates.
tag	A piece of semantic information (metadata) associated with a device or point (entity) for the purpose of filtering or grouping entities. Tags identify the purpose of the component or point and its relationship to other entities. For example, you may wish to view only data collected from meters located in maintenance buildings as opposed to those located in office buildings or schools. For this grouping to work, the metering device in each maintenance building includes a tag that associates the meter with all the other maintenance buildings in your system. JACEs are associated with Supervisors based on tags; searching is done based on tags. Tags are contained in tag dictionaries. Each tag dictionary is referenced by a unique namespace.
tag dictionary	Tag dictionaries contain a set of tag definitions, and may contain tag group definitions, relation definitions, as well as tag rules for smart tags.
template	A deployable package of Niagara objects used to streamline repetitive configuration steps when making multiple installations with similar functionality. For example, when setting up a new device by deploying a

	device template, only unique device properties require configuration. Templates are indexed and searchable.
template designer	The template designer is the person who creates and maintains a template.
template user	The template user is the person who deploys a template to configure an installed instance.

Index

A

application spreadsheet
  editing 53
application template 31
  creating 33
  export configs 52
  installing 34
  upgrading 37
  vs. station template 31
applicationtemplate
  vs.component(device)template 31

B

bulk templates
  deploying via the Template Manager view.. 50, 54

C

Component tab 69
component tree 11
components 59
Components
  Template Service 59
  TemplateConfig 60
Configuration tab 69

D

Deploy a template 23
  via Device Manager 26
  via drag and drop 23
deploy process
  provisioning 65
deployed templates
  provisioning 28
detach deployed template 30
detach process
  provisioning 67
downgrade deployed template 29

E

editing spreadsheet 57
enhancements 7
Excel 46–47, 53
Existing file and component structures for templates 32

F

Find Usage 61

G

Graphics tab 71
graphics views
  configuring in a template 15

I

informational popups 63
I/O links 13

L

lugins 59

M

Managing graphic views 71
module
  creating from a template set 21

N

New Station Wizard 19

O

optional components 39, 57
  establishing 39
  installing 41
  installing bulk application templates 57
overview 7

P

properties
  exposing in a template 12
provisioning
  detach deployed template 30
  downgrade deployed template 29
  redeploy deployed template 29
  upgrade deployed template 28

R

redeploy deployed template 29
relations
  adding to a template 12
Relations tab 70

S

spreadsheet 47
  editing 46
station template
  creating 19
sub-templates
  adding 17
Subtemplate Usage 61
subtemplates 8
  remove 19
Subtemplates tab 72

T

template
  adding relations 12
  bulk deployment 43
  component tree 11
  creating 9
  creating a module 21
  editing 19
  exporting configuration information to an Excel worksheet 44
  exposing properties 12
  making 9
  managing graphic views 15
  managing I/O links 13
  provisioning already deployed templates 28
template configuration
  updating 55
template editor 67
Template I/O tab 71
Template Info tab 67
template inputs
  outputs, relations and configs 48
template instance data 48
Template Manager 62
  batch resolution of unlinked inputs 64
  GoTo navigation 63
  provisioning commands 65
  provisioning deploy process 65
  provisioning detach process 67
  unbound template I/Os 64
  unbound template relations 64
Template Service 59
Template sidebar 60
Template view 67
template.manifest.xml file 74
TemplateConfig 60
templateoptions 73
templates
  updating 18
troubleshooting 59, 75

U

upgrade deployed template 28