

Technical Document

# Provisioning Guide

August 19, 2015

niagara<sup>4</sup>

# Provisioning Guide

## **Tridium, Inc.**

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

## **Confidentiality**

The information contained in this document is confidential information of Tridium, Inc., a Delaware corporation ("Tridium"). Such information and the software described herein, is furnished under a license agreement and may be used only in accordance with that agreement.

The information contained in this document is provided solely for use by Tridium employees, licensees, and system owners; and, except as permitted under the below copyright notice, is not to be released to, or reproduced for, anyone else.

While every effort has been made to assure the accuracy of this document, Tridium is not responsible for damages of any kind, including without limitation consequential damages, arising from the application of the information contained herein. Information and specifications published here are current as of the date of this publication and are subject to change without notice. The latest product specifications can be found by contacting our corporate headquarters, Richmond, Virginia.

## **Trademark notice**

BACnet and ASHRAE are registered trademarks of American Society of Heating, Refrigerating and Air-Conditioning Engineers. Microsoft, Excel, Internet Explorer, Windows, Windows Vista, Windows Server, and SQL Server are registered trademarks of Microsoft Corporation. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Mozilla and Firefox are trademarks of the Mozilla Foundation. Echelon, LON, LonMark, LonTalk, and LonWorks are registered trademarks of Echelon Corporation. Tridium, JACE, Niagara Framework, NiagaraAX Framework, and Sedona Framework are registered trademarks, and Workbench, WorkplaceAX, and AXSupervisor, are trademarks of Tridium Inc. All other product names and services mentioned in this publication that is known to be trademarks, registered trademarks, or service marks are the property of their respective owners.

## **Copyright and patent notice**

This document may be copied by parties who are authorized to distribute Tridium products in connection with distribution of those products, subject to the contracts that authorize such distribution. It may not otherwise, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without prior written consent from Tridium, Inc.

Copyright © 2015 Tridium, Inc. All rights reserved.

The product(s) described herein may be covered by one or more U.S or foreign patents of Tridium.

# Contents

<b>About this guide .....</b>	<b>7</b>
Document change log .....	7
Related documents .....	7
<b>Chapter 1 Provisioning overview.....</b>	<b>9</b>
Provisioning FAQs .....	10
Provisioning steps .....	10
Backup Stations step (N4).....	10
Copy Supervisor File step .....	11
Copy Local File step .....	11
Install Software Step.....	11
Reboot step .....	11
Run Robot step .....	11
Update Licenses step .....	12
Upgrade Out-of-Date Software step .....	12
Provisioning extensions .....	12
Provisioning-related alarms.....	13
Provisioning in a mixed AX/N4 network.....	14
<b>Chapter 2 Provisioning Installation .....</b>	<b>17</b>
Adding the BatchJobService.....	17
Adding the ProvisioningNwExt .....	18
Configuring Platform port and credentials.....	18
<b>Chapter 3 Provisioning job configuration.....</b>	<b>21</b>
One-time jobs .....	21
Creating a one-time job.....	21
Adding one-time job list steps .....	22
Prototype jobs .....	22
Creating a prototype job .....	22
Adding prototype job list steps.....	23
Configuring prototype job retention policy .....	24
Prototype job actions .....	24
Removing Job List Steps.....	25
Reordering Job List Steps.....	25
Adding stations.....	25
Removing Stations .....	26
Reordering stations .....	27
Updating host licenses .....	27
Synchronizing with the Supervisor license database .....	27
Updating licenses from the Network License Summary .....	27
Station backup.....	28
Setting up the backup provisioning job .....	28
Setting up the backup job steps.....	29
Setting up retention policy.....	29
Setting up the prototype job schedule .....	30

- Software installation ..... 30
  - Synchronizing software databases..... 30
  - Installing software using the add button..... 31
  - Installing software by dragging from the Software container ..... 32
- Copying a Supervisor file ..... 33
- Copying a local file ..... 34
- Creating a provisioning robot ..... 35
- Adding a robot step to a provisioning job..... 35
- Chapter 4 Provisioning Job management ..... 37**
  - Job execution ..... 37
  - Provisioning data files..... 37
  - Histories related to provisioning ..... 38
- Chapter 5 Components..... 41**
  - Components in the batchJob module..... 41
    - batchJob-BatchJobService ..... 41
    - batchJob-ThreadPoolJobQueue ..... 42
  - Components in the provisioningNiagara module ..... 42
    - provisioningNiagara-BackupStationExt ..... 43
    - provisioningNiagara-InstallableSummary ..... 45
    - provisioningNiagara-InstallableSpec ..... 45
    - provisioningNiagara-InstallStep ..... 45
    - provisioningNiagara-LicenseStationExt ..... 45
    - provisioningNiagara-PlatformConnection..... 47
    - provisioningNiagara-SoftwareStationExt ..... 48
    - provisioningNiagara-StationProxy ..... 49
- Chapter 6 Plugins (views) ..... 53**
  - Plugins in batchJob module ..... 53
    - Job Log ..... 53
    - batchJob-PrototypeJobList ..... 54
  - Plugins in provisioningNiagara module..... 57
    - provisioningNiagara-BackupStepDetailsView ..... 57
    - Network License Summary ..... 59
    - provisioningNiagara-NiagaraNetworkJobBuilder..... 60
    - Niagara Network Job view ..... 62
    - provisioningNiagara-NiagaraNetworkPrototypeView..... 66
    - provisioningNiagara-ProvisioningManager ..... 67
    - provisioningNiagara-ProvisioningRobotEditor ..... 69
    - provisioningNiagara-ProvisioningStationDirector..... 69
    - provisioningNiagara-StationJobList..... 70
    - provisioningNiagara-StationSoftwareView..... 72
    - provisioningNiagara-SupervisorLicenseManager ..... 73
    - Check stations list ..... 74
- Chapter 7 Troubleshooting..... 77**
  - Why the Start Backup action is NOT recommended..... 77

**Glossary.....79**



## **About this guide**

This guide describes how to set up provisioning jobs on a Niagara Network.

## **Document change log**

Initial release document, August 19, 2015.

## **Related documents**

Following is a list of related guides.

- Niagara 4 Platform Guide
- *Niagara 4 Driver's Guide*





# Chapter 1 Provisioning overview

## Topics covered in this chapter

- ◆ Provisioning FAQs
- ◆ Provisioning steps
- ◆ Provisioning extensions
- ◆ Provisioning-related alarms
- ◆ Provisioning in a mixed AX/N4 network

Niagara provisioning is available only to a Supervisor station. It provides automation of various tasks to remote hosts in the station's **NiagaraNetwork**. For the most part, these are platform tasks—that is, otherwise done using (full) Workbench. Provisioning robot tasks allow the running of custom program code in the host station (executed by each station's **ProgramService**). The Supervisor station automatically performs these tasks, which are modeled in the station as provisioning jobs.

Outside of provisioning, you would have to perform similar tasks on each station using (full) Workbench and one of the following methods:

- Making individual platform connections directly to remote hosts, then using the appropriate platform views.
- Making individual tunneled platform connections to remote hosts, then using the appropriate platform views.
- In the case of provisioning robots, by opening station connections and then copying and executing Program objects.

**NOTE:** For details about the platform user interface, see the Niagara 4 Platform Guide.

Provisioning provides these advantages over individual platform (or station) connections:

- When provisioning, you need only one station connection—to the Supervisor, and no other connections (platform or otherwise). This means that you can run a provisioning job from any location where you can open the Supervisor station...even using Web Workbench! (ordinary platform tasks cannot be done using Web Workbench.)
- Provisioning allows the same series of tasks (executed as job steps), to be run on any number of target hosts. Most job steps execute sequentially on one host, then repeat on the next host, until the tasks are completed on all specified hosts. This ability is useful when performing the same tasks on multiple stations, such as when implementing company-wide software upgrade, or a periodic backup of all hosts' station configurations.
- By default, provisioning provides persistent storage of all jobs on the Supervisor, including all statistics associated with each job and step (creating user, begin and end job times, step details, log output, and so on). In the case of station backups, any saved .dist file can also be restored directly from its batch job step log— via a **Restore** function, whereby it is executed as another provisioning job.

There are two types of provisioning jobs.

- The first is a job intended to be run only once, usually immediately. You build this type of provisioning job using the **Niagara Network Job Builder** which is the default view accessed by double-clicking the **NiagaraNetwork's ProvisioningNwExt** component.
- The second type of job is one that is intended to be run on a regular basis, called a prototype job. This is the type of job you would set up to back up all station configuration data. A prototype job is linked to a **TriggerSchedule**, which specifies when it runs. A prototype job can also be run immediately.

Each job runs one or more provisioning steps:

- Backup Stations (component: **BackupStationExt**)— makes an online backup of each running station.

- Copy Supervisor File (component: `FileCopyStep`) – makes a copy of a Supervisor file.
- Copy Local File (component: `FileCopyStep`) – makes a copy of a local file on your PC.
- Install Software (component: `InstallBySpecStep`) – installs a software module in one or more remote hosts.
- Reboot (component: `RebootJobStep`) – allows you to create custom code to run on each station.
- Run Robot (component: `RunRobotStep`) – reboots the platform host.
- Update Licenses (component: `LicenseStationExt`)– updates the software licenses in each remote host.
- Upgrade Out-of-Date Software (component: `SoftwareStationExt`) – installs software, upgrades out-of-date software, supports copy supervisor step, and the reboot step.

In addition to setting up and running provisioning jobs, provisioning (device) extensions in each station require configuration as well as provide additional provisioning services., under each station modeled in `Drivers→NiagaraNetwork`.

## Provisioning FAQs

Below are some frequently asked questions (FAQs) about the `provisioningNiagara` module:

### What is meant by provisioning?

Provisioning allows an administrator from the Supervisor station to configure the automation of:

- one or more pre-defined tasks
- against one or more (potentially many) stations in the Supervisor’s `NiagaraNetwork`
- done from the Supervisor
- in a way where results are recorded and can be referred to later.

Tasks would otherwise need to be performed manually by a user with Workbench, often by making an individual platform connection to each host. All provisioning work is performed by the Supervisor station, as provisioning jobs.

### What type of pre-defined tasks can provisioning perform?

Provisioning includes the ability to do station backups, install software, update licenses, and copy files, among other things. Also, you can write custom program code to execute as provisioning robots. This feature makes it possible to change the station database.

### Is there any other sort of provisioning besides “provisioning Niagara”?

The information in this document applies only to the Niagara driver, meaning for Niagara hosts represented in the `NiagaraNetwork` of a Supervisor station. However, with the separation of provisioning functions into different modules, reflected by a `BatchJobService` as well as a network-level `ProvisioningNwExt`, other driver types may support provisioning in the future.

## Provisioning steps

Each provisioning job consists of one or more steps that the Supervisor’s `JobService` controls and the remote station’s `ProgramService` executes on the remote host. These steps include station backup, update and file copy, among others.

### Backup Stations step (N4)

This step makes an online station backup if the station is running, or an offline backup if it is idle. The Backup Stations step is available in the `New Job Step` menu when adding a step in either the `Niagara Network Job Builder` or `Niagara Network Prototype View` (preferred) views.

**NOTE:** Configuring a regularly-scheduled station backup by adding a provisioning job prototype (using the **Niagara Network Prototype View**) is a better practice than setting up a one-time station backup (using the **Niagara Network Job Builder**).

The system automatically adds the Backup Stations step when you manually invoke the `Start Backup` (for every station).

The system stores the backup dist file for each station on the Supervisor, using the following convention:

```
^provisioningNiagara/stationData/stationName/backups/backup_stationName_timestamp.dist
```

where

- `stationName` is the name of the station processed in the step
- `timestamp` is the job's start time, formatted as `YYYYMMDD_HHmms.sss`

Super user permissions are required to see the `provisioningNiagara` subfolder on the Supervisor station.

The resulting backup file is encrypted and requires credentials to restore.

## Copy Supervisor File step

This step copies a single file from one location on your the Supervisor PC to a location in all specified stations. It is available in the **New Job Step** menu when adding a step in either the **Niagara Network Job Builder** or **Niagara Network Prototype View**.

When creating a one-time job using the **Niagara Network Job Builder**, you can select either a Supervisor file or a local file on your PC to copy. If you are creating a job prototype using the **Niagara Network Schedule View**, you can select only a Supervisor file.

## Copy Local File step

This step copies a single file from your (local) Workbench PC to a given location on the target host. It is available when adding a step to a one-time provisioning job using the **Niagara Network Job Builder**.

## Install Software Step

This step Installs a versioned software module or dist on the target host(s). It is available in the **New Job Step** menu when adding a step in either the **Niagara Network Job Builder** or **Niagara Network Prototype View**.

## Reboot step

This step (component: `RebootJobStep`) restarts the station's host (platform), then waits until its platform daemon comes back up and is available for connections to begin. It is available in the **New Job Step** menu when adding a step in either the **Niagara Network Job Builder** or **Niagara Network Prototype View**.

Usage is expected to be infrequent, perhaps as a temporary measure for some misbehaving third-party software module. This job step fails if a station's `Software` device extension is disabled or in fault.

## Run Robot step

This step passes the specified `ProvisioningRobot` (custom-created program code) located in the Supervisor station to the `ProgramService` on each remote station. The station's `ProgramService` runs the custom code in the remote station. You access this step in the **New Job Step** menu when adding a step in either the **Niagara Network Job Builder** or **Niagara Network Prototype View**.

The Supervisor station must have a `ProvisioningRobot` customized to perform a task (or tasks) that specifically apply to all stations included in the provisioning job. This job step fails if a target station is not running, does not have the `ProgramService`, or selectively in other cases were faulty or inapplicable program code is included in the specified `ProvisioningRobot`.

As a simple test, specify the `ProvisioningRobot` itself (as copied from the `provisioningNiagara` palette), as a Run Robot step (or by clicking the Run Now button). The `ProvisioningRobot` should execute without errors on any target station with the `ProgramService`.

The program code in this `ProvisioningRobot` includes a number of helpful programming tips within the remark lines near the top of the code.

## Update Licenses step

This step updates the licenses on all remote hosts included in the job.

The Update Licenses step (`UpdateLicensesJobStep`) is the only step option for the **Initial steps to run only once** (top) step list in both the **Niagara Network Job Builder** (used to create one-off jobs) and **Niagara Network Prototype View** (used to create repeating jobs). When processed by the Supervisor, it gathers information about the licenses installed on each target host, then accesses the online licensing server (in one message) to see if the licenses are up-to-date. If a host's license is out-of-date, the step updates each licence in the target station's host, and in the Supervisor's local license database.

**NOTE:** If the Supervisor is not configured for Internet connectivity, only its local license database is used to compare against licenses installed in the target host(s). If a host's license is out-of-date, the step updates the license in the target host (s).

For related information, refer to the Niagara 4 Platform Guide.

## Upgrade Out-of-Date Software step

This step (`UpgradeOutOfDateStep`) compares the versions of software installed on the station's host with the latest versions of the same software in the Supervisor's software database. Any software the step finds with a higher version on the Supervisor it installs to the station. This step is in the **New Job Step** menu when adding a step in either the **Niagara Network Job Builder** or **Niagara Network Prototype View**.

**NOTE:** Although not typical, the latest version of any software module found under the Supervisor's software database (under `!sw`) is always installed by this step, even if the Supervisor itself is using an earlier installed version (as found in its `!modules` directory). Normally, a Supervisor has the latest versions of software modules installed, so this distinction is moot.

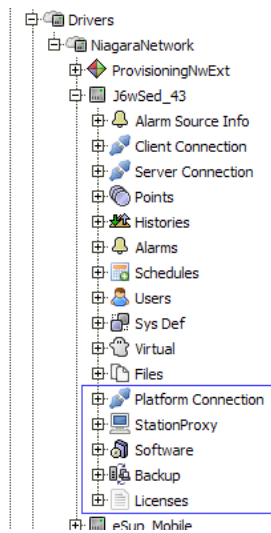
To run more efficiently, the system combines the upgrade out-of-date-software steps with other software install steps and copy file steps.

## Provisioning extensions

This topic introduces the five special device extensions that are automatically created under each of the **NiagaraStation** devices contained in its **NiagaraNetwork**. Provided the network has the `ProvisioningNwExt`, these extensions are also automatically included when you add a new station under the **NiagaraNetwork** in the Supervisor.

The provisioning extensions appear under a different area of the Supervisor station from the station's `ProvisioningNwExt`, and from the job prototype components separately copied from the `provisioningNiagara` palette.

Figure 1 Provisioning extensions added to NiagaraStation device



These five device extensions are in addition to the standard extensions (**Points**, **Histories**, **Alarms**, **Schedules**, **Users**, **Sys Def**, and **Files**) that exist for any **NiagaraStation** device.

Some of the extensions have special views. Along with the **ProvisioningNwExt** and its children, all station extension components are required to be present (and enabled) for full provisioning support.

The five provisioning extensions are:

- **Platform Connection** manages the link between the Supervisor and the remote host running the station.
- **StationProxy** polls the station for system statistics.
- **Software** holds a snapshot of the current software versions installed on the remote host.
- **Backup** enables the Supervisor to make backups of the station.
- **Licenses** enables the Supervisor to update licenses on the host that is running the station.

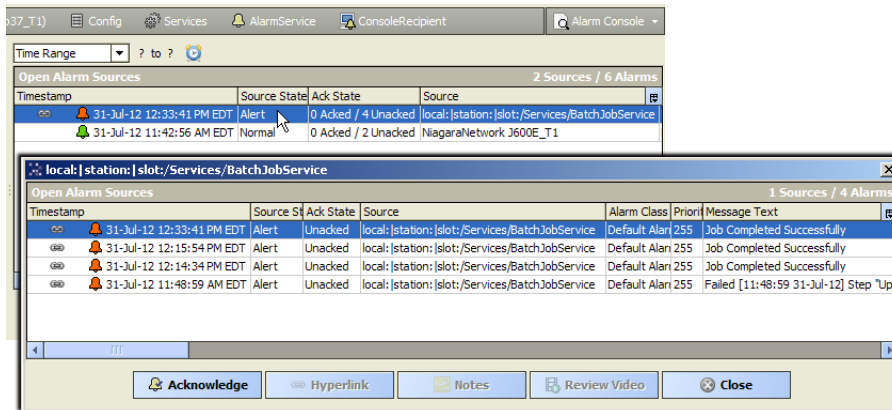
## Provisioning-related alarms

For any provisioning job, you can configure it to generate an alarm either upon job failure, successful job completion, or both.

You configure alarms on the **Niagara Network Job Builder** and **Niagara Network Prototype View**.

Alarm routing uses the alarm class specified in the Supervisor's **BatchJobService** properties, and any alarm appears as an Alert 🚨 (source state) in the alarm console, which identifies the **BatchJobService** ord as source (local:|station:|slot:/Services/BatchJobService).

Figure 2 Niagara Provisioning alarm is an alert with source as BatchJobService



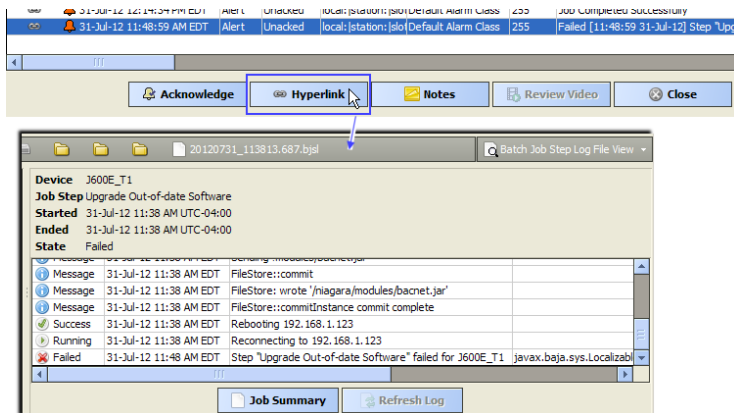
The single alarm source of the **BatchJobService** applies to all provisioning-related alarms (alerts). This means that only one row is used in the alarm console for all provisioning alerts. Keep in mind if there are multiple alerts. When you click **Acknowledge** on that row, you are acknowledging all provisioning alerts.

To see multiple provisioning alerts in the **Alarm Details** view for the **BatchJobService**, double-click the row.

Each provisioning alert includes a hyperlink icon for more detail. When you select (highlight) the alert and click the **Hyperlink** button, the view changes to either:

- **Batch Job Log File View** — For that job, in cases where the alert is raised for successful job completion.
- **Batch Job Step Log File View** — For a failed job step, in cases where the alert is raised for a job failure.

Figure 3 Batch Job Step Log File View from Hyperlink in Alarm Console



If you disposed of a provisioning job before acknowledging it from the **Alarm Console**, and you click the **Hyperlink** button to view it from an associated Alarm Details view, the system advises, “Cannot Display Page.” This happens because disposing of a job removes the batch job log (.bjl) file and all batch job step log (.bjsl) files associated with that job. Therefore, acknowledge any related alarms (alerts) before disposing of provisioning jobs.

## Provisioning in a mixed AX/N4 network

For a number of reasons, including the increased emphasis in Niagara 4.0 on security, limitations apply in installations where hosts running AX-3.8 and others running N4.0 share the same network.

Be aware that:

- An N4.0 Supervisor can provision an AX-3.8 station using these steps: Backup Stations, Copy Supervisor File, Install Software, Update Licenses, and Upgrade Out-of-Date Software.
- An AX-3.8 Supervisor cannot provision an N4.0 station.
- N4.0 provisioning robots do not run on AX-3.8 stations.





# Chapter 2 Provisioning Installation

## Topics covered in this chapter

- ◆ Adding the BatchJobService
- ◆ Adding the ProvisioningNwExt
- ◆ Configuring Platform port and credentials

Provisioning software requires multiple modules installed in the modules folder of the Supervisor PC.

### Supervisor requirements:

- NiagaraAX-3.5 or later is installed on your Supervisor.
- The Supervisor must be licensed for provisioning. A licensed Supervisor has the following entry in its `Tridium.license (license) file`:

```
<feature name="provisioning"
expiration="never"
parts="ENG-WORKSTATION"/>
```

- The following modules must be installed in the modules folder of the Supervisor PC.

- provisioningNiagara
- batchJob

### Host controller requirements:

- Controllers should be running the same Niagara release as the Supervisor, for example Niagara 4.0. However, hosts running an earlier release than the Supervisor may be supported for most provisioning operations.
- Controllers do not require any provisioning-related module (provisioningNiagara, batchJob, provisioning), nor do they require a license feature for provisioning. They do require the program module (and the **ProgramService** in their station) in order to process provisioning jobs with Run Robot steps.

## Adding the BatchJobService

The **BatchJobService** manages provisioning jobs.

**Prerequisites:** The Supervisor PC has been licensed for provisioning and has the required modules installed.

- Step 1 If it is not already open, in Workbench, open the Supervisor station.
- Step 2 Open the **provisioningNiagara** palette in the Workbench palette side bar.
- Step 3 Expand the station's **Config** space to see its **Services** folder, and double-click it for the **Service Manager** view.
- Step 4 Do one of the following:
  - If a **BatchJobService** is already listed, verify that it is enabled (status `ok`).
  - If no **BatchJobService** exists, from the palette, drag the **BatchJobService** onto the station's **Services** folder. In the popup **Name** window, rename the service—or use the default name and click **OK**.A **BatchJobService** (or whatever you named it), appears under your **Services** folder.
- Step 5 Right-click the **BatchJobService** and click `Property Sheet`.
- Step 6 Set the service's **Job Queue**→**Max Threads** property to a value greater than 1, say 2 or 3.

This allows more than a single provisioning job to run concurrently.

- Step 7 To monitor provisioning alarms, change the **BatchJobService Alarm Class** from the default `Default Alarm Class` to another alarm class

## Adding the ProvisioningNwExt

**Prerequisites:** The Supervisor PC has been licensed for provisioning and has the required modules installed.

- Step 1 If it is not already open, in Workbench, open the Supervisor station.
- Step 2 Open the **provisioningNiagara** palette in the Workbench palette side bar.
- Step 3 Under the station's **Config** space, expand its **NiagaraNetwork**.
- Step 4 If a **ProvisioningNwExt** is already a child of the network, verify that it is enabled (status `ok`), and skip ahead to the next procedure, .
- Step 5 From the palette, drag the **ProvisioningNwExt** onto the station's **NiagaraNetwork**. In the pop-up **Name** window, you can rename the **ProvisioningNwExt**—or, use the default name and Click **OK**.

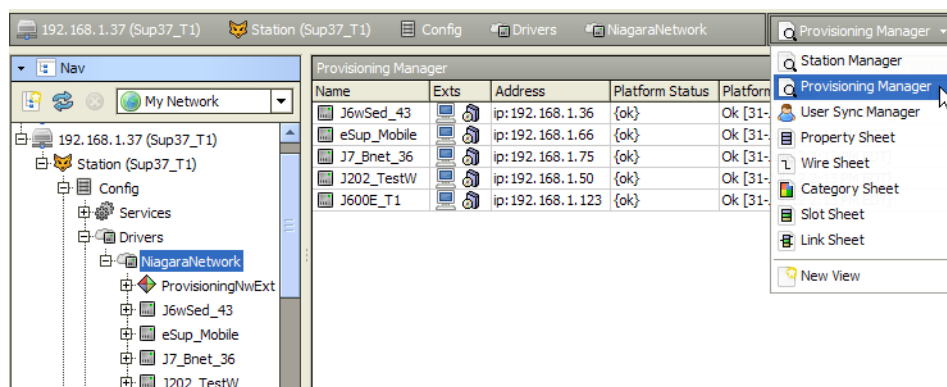
You now have a **ProvisioningNwExt** (or whatever you named it), under your **NiagaraNetwork**. When you add this extension to a Supervisor station, every existing **NiagaraStation** device (under the network) automatically receives five new device extensions. As you add more **NiagaraStations**, they too have these extensions

## Configuring Platform port and credentials

For each **NiagaraStation** device in the Supervisor station's **NiagaraNetwork**, you must confirm the port and enter the corresponding platform daemon credentials of its host. The simplest way to do this is by using either the **Station Manager** or **Provisioning Manager** views.

Configuring platform credentials is especially useful if multiple (if not all) remote hosts are using the same platform credentials. In this case, a single gang edit of these selected credentials provides the configuration needed for Supervisor access.

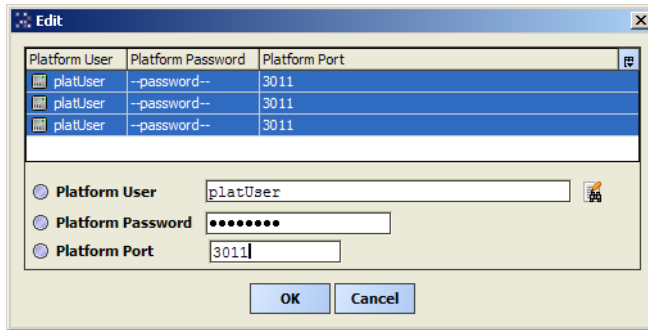
- Step 1 In the Nav tree, expand **Config**→**Drivers** and right-click the and click the **Station Manager** or **Provisioning Manager** actions.



- Step 2 Select one or more stations to edit.

- Step 3 Right-click and select **Edit**.

The system opens the **Provisioning Manager Edit** window.



The last few **Platform** fields are for editing the station's platform connection extension properties, namely for platform **User**, **Password**, and **Port**. In AX-3.7 and AX-3.8 versions, an additional **Secure Platform** boolean property is preset to `false` by default. In N4.0 versions the **Secure Platform** boolean property must be `true`.

**Step 4** If needed, set **Secure Platform** to `true`.

Notice that the default platform port is not 3011, but is 5011—unless changed to some other port for the specific platform.

**Step 5** Enter platform connection credentials, and, if using a non-default port address, change the port from the default port 5011 to the port number used.

Your platform credentials and port should be the same credentials and port you use when you open a platform connection directly to this remote host.

**Step 6** To continue, click **OK**.

Upon the next monitor ping of the host's platform daemon, its **Station Run State** should change from `Unknown` to `Running` and the station's **Platform Connection** extension should now have a status of `ok`. To ping the platform daemon immediately, right-click a **NiagaraStation's Platform Connection** extension and click the action `Ping`.

**Step 7** Confirm that the station, as shown in the **Provisioning Manger**, has a platform status of `ok`.

**Step 8** Test the validity of the credentials, right-click the station row and select `Ping` from the list.

Repeat this procedure for each **NiagaraStation** listed in the **Station Manager** view.



# Chapter 3 Provisioning job configuration

## Topics covered in this chapter

- ◆ One-time jobs
- ◆ Prototype jobs
- ◆ Removing Job List Steps
- ◆ Reordering Job List Steps
- ◆ Adding stations
- ◆ Removing Stations
- ◆ Reordering stations
- ◆ Updating host licenses
- ◆ Station backup
- ◆ Software installation
- ◆ Copying a Supervisor file
- ◆ Copying a local file
- ◆ Creating a provisioning robot
- ◆ Adding a robot step to a provisioning job

The benefit of using a provisioning job is that you can run multiple steps against one or more remote hosts.

There are two types of provisioning jobs:

- One-time provisioning jobs is for executing special, infrequent tasks, such as updating license files or software modules in selected (or all) remote hosts. A one-time provisioning job uses the **Niagara Network Job Builder** (the default view of the **ProvisioningNwExt**) to configure sequences of provisioning steps against one or more stations in a Supervisor's **NiagaraNetwork**.

Unlike prototype jobs, a one-time provisioning job cannot be saved as a component for reuse.

- Regularly scheduled provisioning jobs are especially useful for large enterprises with hundreds of remote hosts. For this type of job you create and configure a **NiagaraNetworkJobPrototype** component, which can be saved and reused. The job prototype allows you to configure job retention, which is not available when creating a one-time job.

A regularly scheduled provisioning job is especially useful for large enterprises with hundreds of remote hosts. For this type of job you create and configure a **NiagaraNetworkJobPrototype**, which can be saved and reused. A prototype job allows you to configure job retention, which is not available when creating a one-time job. For job retention purposes you should run a backup-stations step only using a prototype job.

## One-time jobs

The **Niagara Network Job Builder** is the default view on the **ProvisioningNwExt**. You start here to specify a one-time provisioning job needed now, meaning that you do not need to save it as a reusable component.

If you would rather build a job that you can schedule or run at a later time, save, duplicate and modify, and so on, use a **NiagaraNetworkJobPrototype** component copied anywhere in the Supervisor's station (each has its own equivalent view). Find them in the **provisioningNiagara** palette. In general, those are the components you should use to create regular station backup jobs.

## Creating a one-time job

This general procedure uses the **Niagara Network Job Builder** to create a one-time provisioning job.

**Prerequisites:** The Supervisor is licensed for provisioning Niagara and the BatchJobService and ProvisioningNwExt components are available under your NiagaraNetwork.

Step 1 Double-click **ProvisioningNwExt**.

The system displays the **Niagara Network Job Builder** view.

Step 2 In the top **Initial steps to run only once** pane, click the **Add** button.

Step 3 In the **New Job Step** popup window, click the step to add and click **OK**.

Step 4 In the lower **Stations to include in the job** pane, click the **Add** button,

Step 5 In the **Add Device** popup window, click to select the stations and click **OK**.

Step 6 To initiate the provisioning job, review your choices and click the **Run Now** button at the bottom of the **Niagara Network Job Builder View**.

The view changes to the **Niagara Network Job View**, where steps and results appear as they are executed.

## Adding one-time job list steps

These job list steps define the one-time tasks to be performed on one or more stations. The **BatchJobService** provisions each station following the sequence defined in the job list steps.

Step 1 Click the + (add) button below the list.

The system displays the **New Job Step** menu.

Step 2 Choose the step type from the **New Job Step** menu.

The same types of job steps are available as in the **Niagara Network Prototype View**.

Step 3 Select the copy action: **Copy Local File** or **Copy Supervisor File**.

Step 4 Right-click in the steps list, select **Add**, and choose the step type from the menu.

Step 5 Drag a file from Workbench's Nav tree into the steps list (implicit File Copy step).

Step 6 Drag a software item (module or .dist) from Workbench's Nav tree that appears under the **ProvisioningNwExt's Software** container, into the **Job Steps List** (implicit Install Software step).

Step 7 Drag a **ProvisioningRobot** that exists in the station's **Config** (component) architecture into the **Job Steps List** (implicit Run Robot step).

## Prototype jobs

A regularly scheduled provisioning job is especially useful for large enterprises with hundreds of remote hosts. For this type of job you create and configure a **NiagaraNetworkJobPrototype**, which can be saved and reused. A prototype job allows you to configure job retention, which is not available when creating a one-time job.

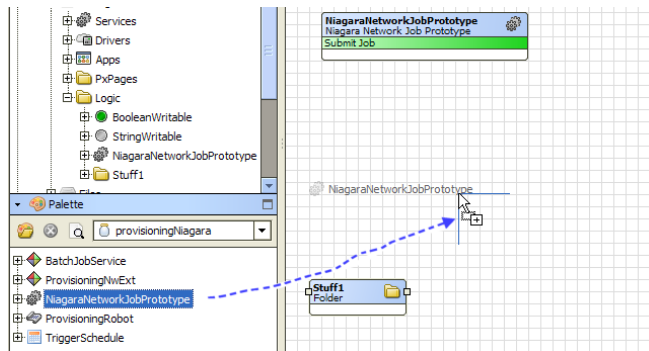
### Creating a prototype job

The benefit of creating a prototype job is that you can specify various provisioning steps against one or more stations in the **NiagaraNetwork** of the Supervisor station and save them as normal persisted components. You can duplicate and edit them as needed. Each **NiagaraNetworkJobPrototype** provides a default **Niagara Network Prototype View** for managing job steps. A prototype job is the recommended method for regular, scheduled backups of networked controllers.

**Prerequisites:** The supervisor is licensed for provisioning Niagara and the BatchJobService and ProvisioningNwExt components are available under your NiagaraNetwork.

Step 1 Copy a **NiagaraNetworkJobPrototype** component for each station from the **provisioningNiagara** palette into the Supervisor station.

You may locate **NiagaraNetworkJobPrototype** components anywhere needed in the architecture of the Supervisor's station database. They do not need to be under the **ProvisioningNwExt**, or even the **NiagaraNetwork**.



The screen capture shows a job prototype (**NiagaraNetworkJobPrototype** component) added from the palette.

**Step 2** Double-click the component to open its **Niagara Network Prototype View**.

The prototype job is ready to add steps.

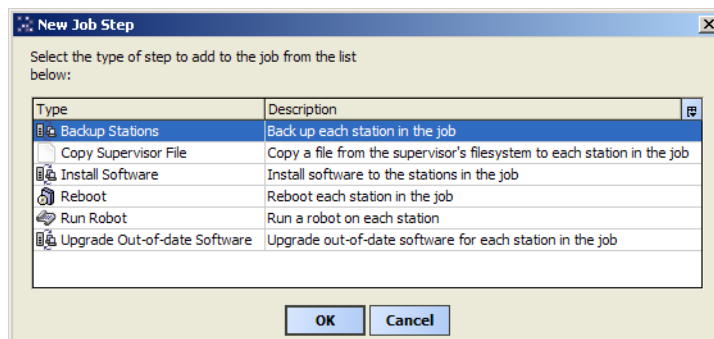
## Adding prototype job list steps

A prototype job is meant to be processed against one or more stations on a regularly scheduled basis. The steps to perform are very similar to those for creating a one-time provisioning job. For clarity, they are separated into separate topics.

**Prerequisites:** The **Niagara Network Prototype View** for the prototype job is open.

**Step 1** Click the + (add) button below the list.

The system displays the **New Job Step** menu.



**Step 2** Choose the step type from the **New Job Step** menu.

The same types of job steps are available as in the **Niagara Network Job Builder**, with the exception of the **Copy Local File** step.

**Step 3** Right-click in the steps list, select **Add**, and choose the step type from the menu.

**Step 4** Drag a file from Workbench's Nav tree into the steps list (implicit File Copy step).

**Step 5** Drag a software item (module or .dist) from Workbench's Nav tree that appears under the **ProvisioningNwExt's Software** container, into the **Job Steps List** (implicit Install Software step).

**Step 6** Drag a **ProvisioningRobot** that exists in the station's **Config** (component) architecture into the **Job Steps List** (implicit Run Robot step).

The prototype job is ready for the `BatchJobService` to provision each station following the sequence defined in the job list steps.

## Configuring prototype job retention policy

Among the various types of batch job components, the ability to set up the disposal of information related to a previously executed job is unique to a job prototype component. When a batch provisioning job is disposed of, any associated files stored on the Supervisor are deleted, and the job removed from the various job list views.

Disposal is especially important for jobs with backup steps, each of which results in an associated backup .dist file to be stored for each station. Although a Supervisor platform has large amounts of hard disk storage, over time, retaining all .dist files can consume huge amounts of file space. As a best practice, you should periodically back up all files to removable media and set the job retention to automatically delete old backups.

By default (as copied from the `provisioningNiagara` palette), a job prototype component's retention is set for all of its jobs to be permanently retained (until manually disposed of). However, you can (and often should) modify a job prototype's retention policy such that some executed jobs are automatically disposed of—depending on either elapsed time or by reaching some number of job executions.

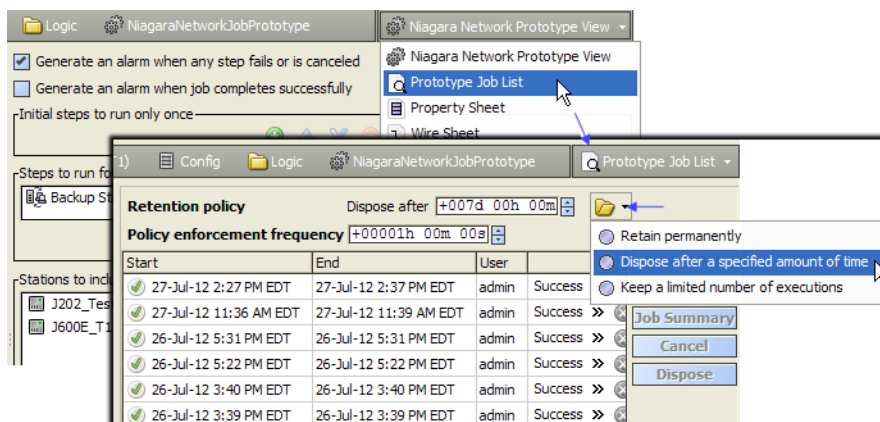
You can access the job retention properties from either the **Prototype Job List** view (this procedure) or from the component's property sheet.

**Step 1** Double-click the `NiagaraNetworkPrototypeJob` container.

The system displays the **Niagara Network Prototype View**.

**Step 2** Click the drop-down list in the upper right corner of the view and click `Prototype Job List`.

The system displays the **Prototype Job List**.



**Step 3** Click the file folder to the right of the **Retention policy** property and select the retention period.

**Step 4** Based on your selection, fill in the **Dispose after** or **Executions to retain** properties.

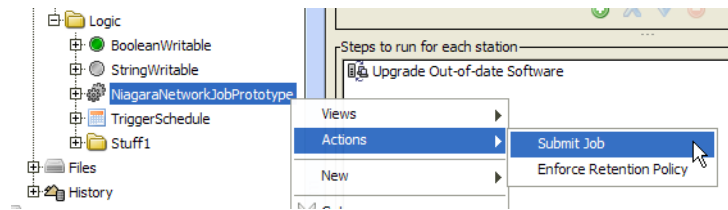
**NOTE:** Keep in mind that if you configure retention properties and then duplicate the job prototype for the purpose of creating a new provisioning job, all copied job components will have that same retention configuration.

## Prototype job actions

Each job prototype (provisioning job) has two available right-click actions.



Figure 4 Actions for a NiagaraNetworkJobPrototype



The **Submit Job** action is equivalent to the **Run Now** button when in the job prototype's default **Niagara Network Prototype View**. The **Enforce Retention Policy** action immediately applies the job's retention policy.

## Removing Job List Steps

Remove a provisioning job step using either of these two methods:

- Click to select the step, then click the **X** (remove) button below the list.
- Right-click the step, and select **Remove** from the popup menu.

## Reordering Job List Steps

You may reorder a selected job step using either of these two methods.

- Click the (up) or (down) arrow button at the bottom of the list.
- Right-click a job step and click **Move Up** or **Move Down**, as needed.

The **JobService** executes job steps from top to bottom in the order that you define them in the two step lists: initial steps first, then steps for each station.

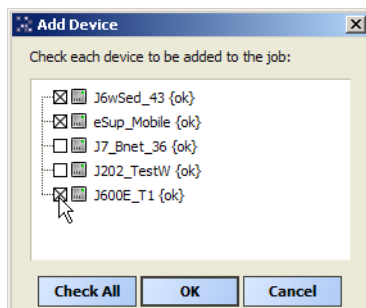
## Adding stations

You can add one or more stations at the same time. This procedure works for adding stations to a one-time job a prototype job.

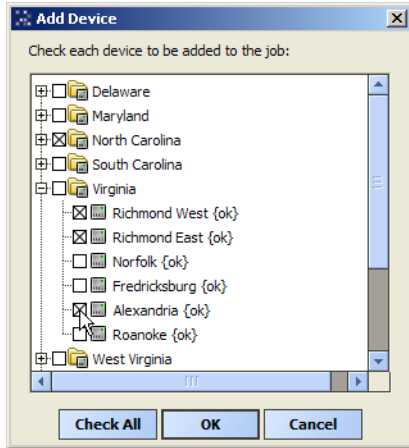
**Prerequisites:** All NiagaraStations are properly configured for platform connections.




Step 1 Do one of the following:

- Click the **+** (add) button below the list, choose the station(s) in the **Add Device** window and click **OK**. Use the **Check All** button to choose all stations.

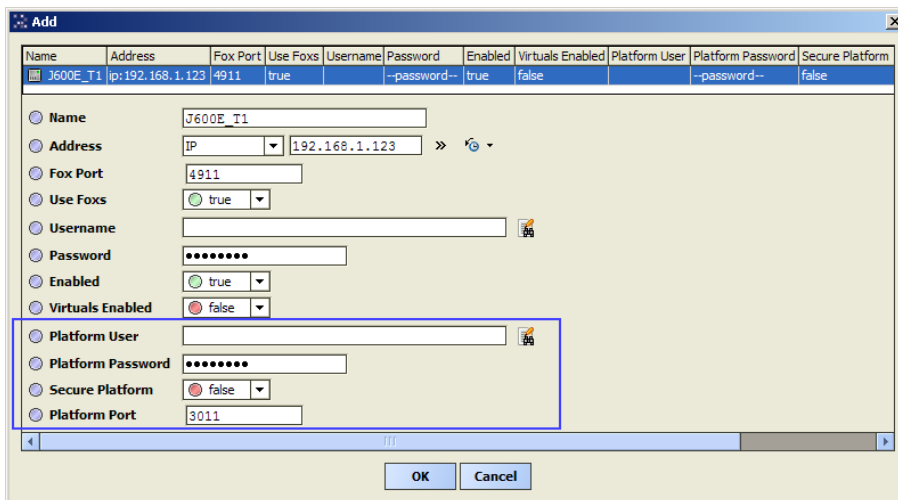


- Right-click in the **Stations List**, select **Add**, and choose the station(s) in the same window.
- From the Nav tree, drag a station from under the **NiagaraNetwork** into the **Stations List**.



This station tree structure using station display names (if assigned) can be useful on Supervisors that have very large numbers of subordinate host controllers. Click on the controls to expand  and collapse  station folders, as needed. A selected  folder will select all nodes under it (stations and any subfolders).

The **Add** window opens for each station.



At the bottom of the **Add** window are three provisioning-related properties.

**Step 2** Enter values for these properties. Always set **Secure Platform** to `true`.

**NOTE:** A NiagaraNetwork discover can determine if a station is configured for secure (Foxs) access, including the port used—for example the standard 4911. However, a discover cannot determine if the **Secure Platform** property is enabled or what the associated port is on the station's host.

## Removing Stations

You may remove a station from a job in the **Niagara Network Job Builder** or **Niagara Network Prototype View** using either of these methods.

- Click to select the station, then click the **X** (remove) button below the list.
- Right-click the station, then select **Remove** from the popup menu.

## Reordering stations

Reorder a selected station in the **Niagara Network Job Builder** or **Niagara Network Prototype View** using one of two methods.

- Click the (up) or (down) arrow button at the bottom of the list.
- Right-click a station and select **Move Up** or **Move Down**, as needed.

Stations are processed in the same top-to-bottom order as defined in the Stations List.

## Updating host licenses

This procedure uses the **Niagara Network Job Builder** to create a one-time provisioning job that updates one or more host licenses in a network.

**Prerequisites:** The BatchJobService and ProvisioningNwExt components are available under your NiagaraNetwork.

Step 1 Double-click **ProvisioningNwExt**.

The system displays the **Niagara Network Job Builder** view.

Step 2 In the top **Initial steps to run only once** pane, click the **Add** button.

Step 3 In the **New Job Step** popup window, click the **Update Licenses** step and click **OK**.

Step 4 In the lower **Stations to include in the job** pane, click the **Add** button,

Step 5 In the **Add Device** popup window, click to select the stations and click **OK**.

Step 6 To initiate the provisioning job, review your choices and click the **Run Now** button at the bottom of the **Niagara Network Job Builder View**.

The view changes to the **Niagara Network Job View**, where steps and results appear as they are executed.

The licenses for all selected hosts are up-to-date. To make updating licenses a regular, automatic event, you need to create a job prototype.

## Synchronizing with the Supervisor license database

The local Supervisor maintains a database that includes information about each host's license. Periodically, it is a good idea to interrogate each host and update the Supervisor's license database.

**Prerequisites:** You have a network of licensed hosts.

Step 1 Expand the **ProvisioningNwExt** in the Nav tree to see its **Licenses** node.

Step 2 Right-click **Licenses** and select **Views→Supervisor License Manager**.

The **Supervisor License Manager** window opens.

Step 3 Click **Synchronize**.

The system prompts with the option to **Synch All Licenses?**

Step 4 Click **Yes** and, at the **Synchronization Complete** prompt, click **OK**.

The Supervisor's license database contains the license identifier for each host in the network.

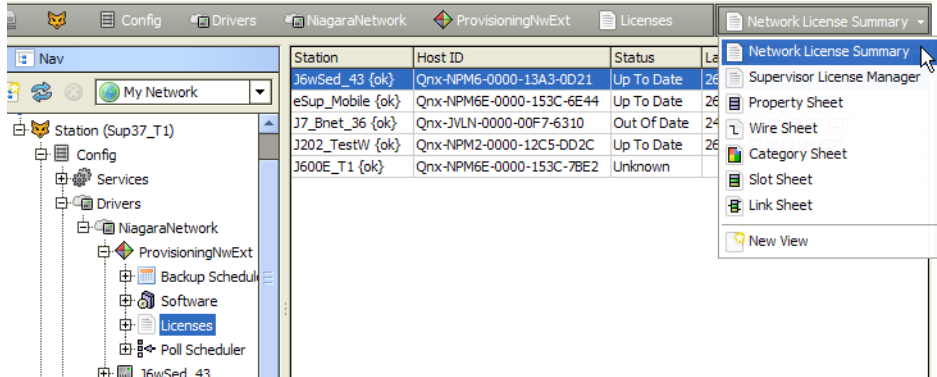
## Updating licenses from the Network License Summary

Rather than create a one-time provisioning job, you can update the license on one or more remote hosts using the **Network License Summary**.

**Prerequisites:** You have synchronized the Supervisor's license database with the host controllers in your network, purchased a license upgrade for each host, and the upgrades are available on the online licensing server.

**Step 1** Select the **Licenses** slot on the **ProvisioningNwExt**.

The system displays the **Network License Summary**.



**Step 2** Select one or more stations and click **Update**.

If a newer license is found (than that already installed), the system installs it in the remote host (s), updates the license(s) in the Supervisor's local license database, and resets the **Last Updated** timestamp to the time of the update.

## Station backup

Configuring a job prototype to back up all stations provides a good example of how to set up a provisioning job.

You can use the **BackupSchedule** slot of the **ProvisioningNwExt** to specify a regular scheduled backup for all stations. Or, you can use job prototype components copied from the **provisioningNiagara** palette and pasted in the station.

The topics that follow provide an example procedure for regularly backing up a Supervisor station with 36 device stations in its **NiagaraNetwork**.

## Setting up the backup provisioning job

Setting up a provisioning job begins with setting up the necessary components under the station **Config** folder: **ProvBackups** folder, **NiagaraNetworkJobPrototype** component, **TriggerSchedule** component

**Prerequisites:** All stations are properly configured for platform connections. You have super user permissions.

- Step 1 With the Supervisor station opened in Workbench, expand it to reveal its **Config** node.
- Step 2 Right-click **Config** and select **New**→ **Folder**.
- Step 3 Name the folder **ProvBackups** (or whatever name you wish. You can create this folder anywhere under **Config**.)
- Step 4 Open the **Palette** side bar, and, in the side bar, open the **provisioningNiagara** palette.
- Step 5 From the **provisioningNiagara** palette, drag a **NiagaraNetworkJobPrototype** component to the new **ProvBackups** folder.
- Step 6 From the **provisioningNiagara** palette, drag a **TriggerSchedule** component to the same **ProvBackups** folder.

Step 7 To view its wire sheet, double-click the **ProvBackups** folder.

Step 8 In the wire sheet, link the **Trigger** slot of the **TriggerSchedule** to the **SubmitJob** slot of the **NiagaraNetworkJobPrototype** component.

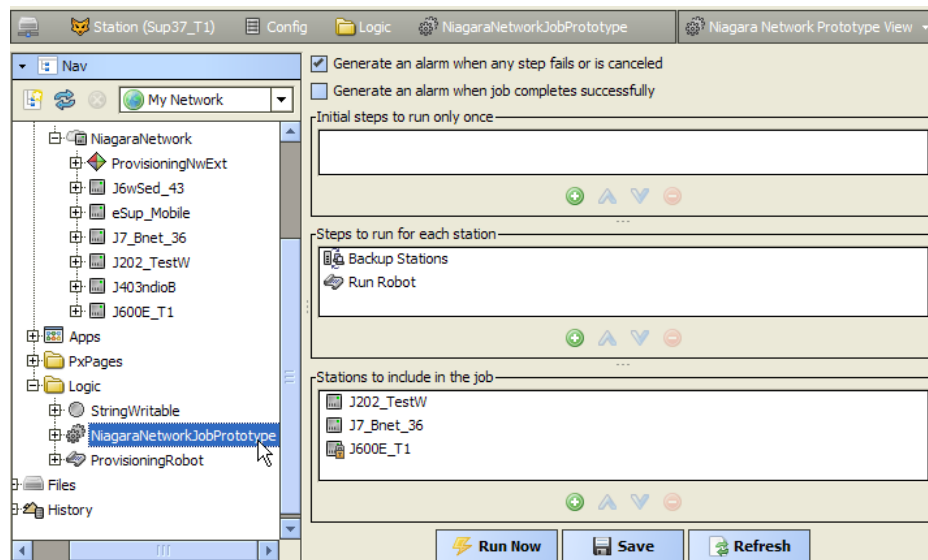
## Setting up the backup job steps

Each action to be taken is a step in the provisioning job.

**Prerequisites:** The station Config container contains the necessary folder and components.

Step 1 To view the provisioning job's **Niagara Network Prototype View**, double-click the **NiagaraNetworkJobPrototype**.

The system displays the view.



Step 2 Configure how you want failed and completed jobs to appear on the alarm console.

**NOTE:** The alarm check box settings only apply to the provisioning job being built, and do not affect other provisioning jobs that may already exist either as other provisioning components, jobs already queued to run, or built in the Niagara Network Job Builder (one-time jobs).

Step 3 In the middle **Steps to run for each station** pane, click the **Add** button.

Step 4 Select the stations to backup (the **Backup Stations**) and click **OK**.

Step 5 In the bottom **Stations to include in the job** pane, click the **Add** button.

Step 6 In the **Add Devices** window, select 12 stations and click **OK**.

**NOTE:** Later, after duplicating this component and reconfiguring, you will select different stations to backup.

Step 7 Click the **Save** button at the bottom of the view.

## Setting up retention policy

Retention policy determines how long to save the job prototype.

Step 1 For the same **NiagaraNetworkJobPrototype**, go to its (secondary) **Prototype Job List** view.

Step 2 Near the top of its **Prototype Job List** view, click the **Retention policy** control for a drop-down list, and choose either:

- Dispose after a specified amount of time (default is 7 days)

- Keep a limited number of executions (default is 10)

Step 3 Click the **Save** button at the bottom of the view.

## Setting up the prototype job schedule

Although you can manually run a provisioning job, most provisioning jobs run automatically based on a trigger schedule. This example procedure demonstrates how to configure each of the three job prototypes to backup a total of 12 different stations per provisioning job. This includes adjusting each **TriggerSchedule** to a slightly different time of execution. For example, you may have one backup provisioning job scheduled at 12:30AM, another at 1:00AM, and another at 1:30AM.

**Prerequisites:** All stations are properly configured for platform connections.

Step 1 To go to the **Trigger Schedule** view, double-click the linked **TriggerSchedule** and add a new event on the left side, for example: Type: **Week and Day: Sat, Any Week, Any Month** (for a weekly backup), and on the right side add an "off hours" time, for example: 12:30AM.

Step 2 Click to **Save** this schedule configuration.

Later, after duplicating this component and reconfiguring, you will select a slightly different off-hours time.

Step 3 With the **NiagaraNetworkJobPrototype** linked to the **TriggerSchedule**, select both components, then right-click and select **Duplicate**. This puts another linked pair on the wire sheet.

Step 4 Move the copied linked pair to a new spot on the wire sheet, and repeat this procedure again.

## Software installation

Installing software on multiple remote hosts using provisioning involves installing the modules to the Supervisor PC, synchronizing Workbench and Supervisor databases, setting up a one-time provisioning job, and executing the job.

You need to obtain the necessary license(s) for the software, and download the modules into your Supervisor PC.

### Synchronizing software databases

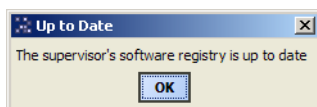
Installing Workbench on a Supervisor PC or engineering workstation makes all licensed software modules available. To update the modules in a remote host using the Supervisor station, you must first synchronize the Workbench and Supervisor databases.

Step 1 In the Supervisor station, open the **Supervisor Software Manager**.

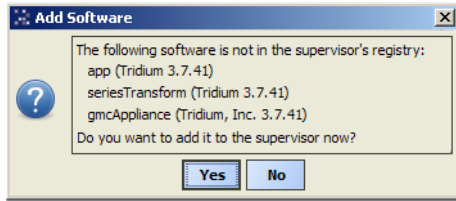
Step 2 Click the **Sync Workbench** button at the bottom of the window.

One of the following happens:

- If the Supervisor already has all the software installable files that are available in the Workbench database, a popup window displays:



- If installable files are available in Workbench that the Supervisor does not have, an **Add Software** window lists the files, and asks if you wish to transfer them to the Supervisor.




Step 3 Click **Yes** or **No** depending on your needs.

### Installing software using the add button

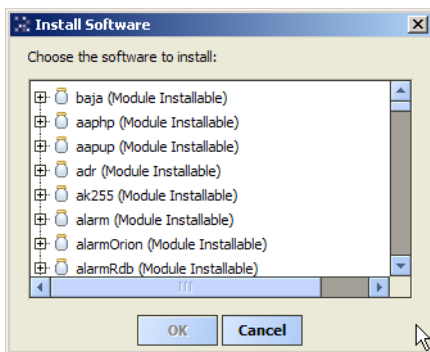
When it is time to install new software on all hosts in a network, you can use provisioning to speed the process of upgrading each remote host. This procedure uses the add button.

**Prerequisites:** The **Niagara Network Job Builder** (for a one-time job) or the **Niagara Network Prototype View** (for a prototype job) is open.

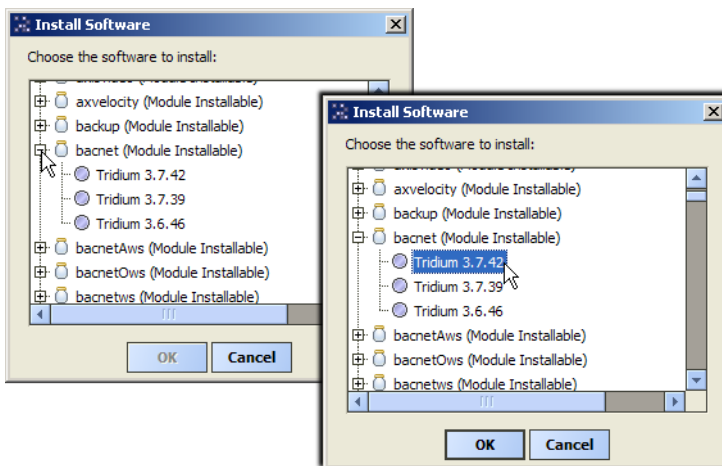
Step 1 If you are using a remote Workbench PC, connect to the Supervisor station.

Step 2 Click the  (add) button to add the Install Software step to the job.

The system opens an **Install Software** window in which you select the software module or .dist file to install.



Step 3 Expand each row by clicking the plus (+) to the left of the module name. This displays the available module version numbers.



Step 4 Select a version and click **OK**.

Once added, the Install Software step appears in the job steps list pane.

To satisfy dependencies, if the software has dependencies on one or more modules that are not yet installed on a particular host, and the modules are in the Supervisor's software registry, the step automatically includes the modules in the processing for host (station).

**NOTE:** It is your (provisioning user's) responsibility to ensure that platform dependencies of the software are met by the hosts running the target stations. For example, it is permissible to have a job with an Install Software step that includes stations running on different platform types, such as JACE-545s and JACE-403s. However, if a step installs a distribution file specific to a JACE-403, note that the dependency check will fail on the JACE-545s, and no software will be installed on JACE-545 hosts.

A slightly different step (InstallStep) is created when you copy/drag a backup .dist file into the **Job Steps List** pane. A backup .dist is not a versioned install (nor is it a FileCopyStep).

To run the provisioning job more efficiently, the system combines Install Software steps with other software install steps, copy file steps, and upgrade out-of-date-software steps.

## Installing software by dragging from the Software container

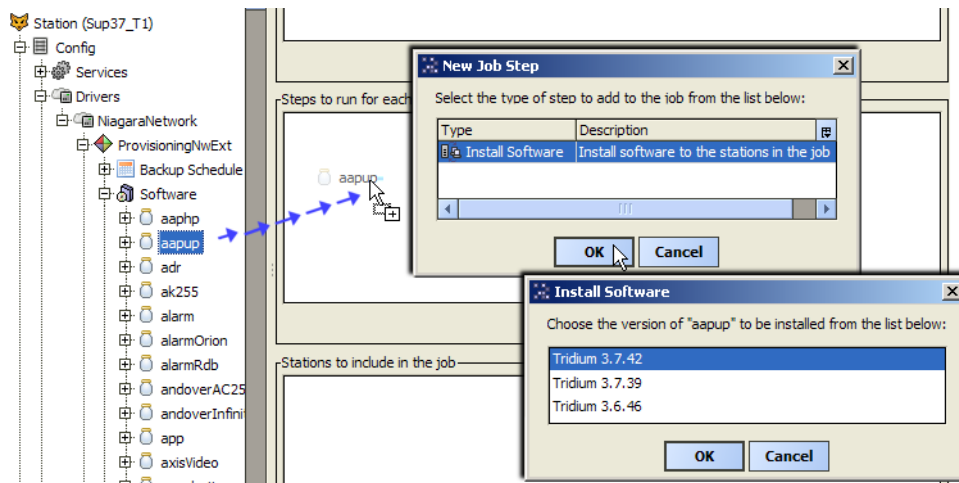
Dragging a file to the Network Job Guider or Niagara Network Prototype View is an easy way of setting up a copy step.

**Prerequisites:** The Niagara Network Job Builder (for a one-time job) or the Niagara Network Prototype View (for a prototype job) is open.

- Step 1 If you are using a remote Workbench PC, connect to the Supervisor station.
- Step 2 On the Supervisor station, locate the new software module(s) in the **Software** container.
- Step 3 From the Nav tree, copy or drag a local software module into the job's step list pane.

The **JobService** automatically checks to see if the software file and version already exist on the Supervisor. If not, the service downloads the file and registers it with the Supervisor. This download process occurs in the background.

If more than one version (file) for the item is in the Supervisor's software database, a popup window prompts you to select the version.



- Step 4 Select the version and click **OK**.

Once added, the Install Software step appears in the job steps list pane.

To satisfy dependencies, if the software has dependencies on one or more modules that are not yet installed on a particular host, and the modules are in the Supervisor's software registry, the step automatically includes the modules in the processing for host (station).



**NOTE:** It is your (provisioning user's) responsibility to ensure that platform dependencies of the software are met by the hosts running the target stations. For example, it is permissible to have a job with an Install Software step that includes stations running on different platform types, such as JACE-545s and JACE-403s. However, if a step installs a distribution file specific to a JACE-403, note that the dependency check will fail on the JACE-545s, and no software will be installed on JACE-545 hosts.

A slightly different step (InstallStep) is created when you copy/drag a backup .dist file into the **Job Steps List** pane. A backup .dist is not a versioned install (nor is it a FileCopyStep).

To run the provisioning job more efficiently, the system combines Install Software steps with other software install steps, copy file steps, and upgrade out-of-date-software steps.

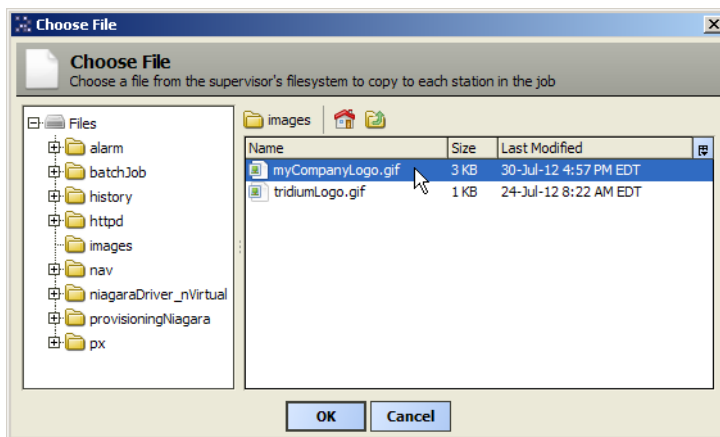
## Copying a Supervisor file

A Supervisor file is a file located in the My File System of the Supervisor's Nav tree.

**Prerequisites:** You have created a new job (one-time, or job prototype) and have either the **Niagara Network Job Builder** or **Niagara Network Schedule View** open.

**Step 1** In the Initial steps to run only once box, click the add button (+) and select the CopyFile step.

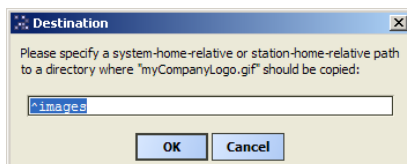
The system opens the file chooser for the files available as part of the Supervisor station.



**Step 2** Using the file chooser, navigate to the source file for the copy operation, select the file, and click **OK**.

If you are creating a one-time job using the **Niagara Network Job Builder**, you can select either a Supervisor file or a local file on your PC. If you are creating a job prototype using the **Niagara Network Prototype View**, you can select only a Supervisor file.

The system prompts you for the target destination.

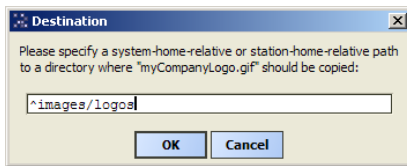


This destination folder applies to all stations in the job. You may need to edit the destination.

**Step 3** Edit the destination system-home-relative or station-home-relative.

The destination string must always begin with the character for either the system-home relative (!) or the station-home relative (^). The system provides no means to modify any files outside of the software release directory on any target host.

For example, your destination folder might be:



In the above example, the destination is changed to specify a `logos` subfolder under the `images` folder, which is at the station root absolute (^). If a destination folder does not already exist on the target host, the step creates it.

#### Step 4 Click **Run Now**.

To run more efficiently, the system combines this step with other file copy steps, install software steps, or upgrade out-of-date software steps in the job.

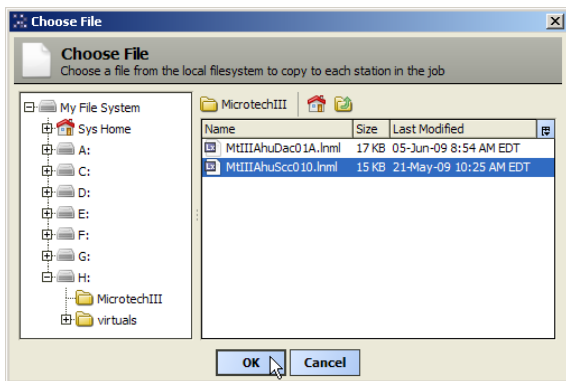
## Copying a local file

A local file may be anywhere on your PC. It is a file that is outside of the directory structure that is available from inside your Supervisor station.

**Prerequisites:** You have created a new one-time job and have the **Niagara Network Job Builder**

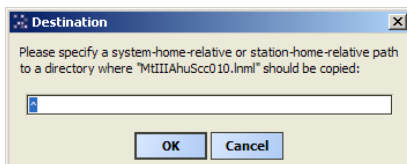
**Step 1** In the Initial steps to run only once box, click the add button (+) and select the CopyFile step.

The system opens the file chooser showing all drives mapped on your local PC.



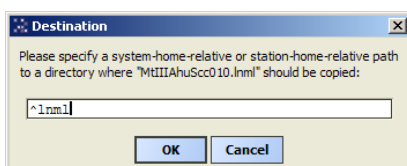
**Step 2** Select a local source file to copy.

A **Destination** window prompts you for the target destination to copy this file to.



This destination folder applies to all stations in the job.

**Step 3** To edit the destination, click into the entry box and type a destination.



In the example above, the destination was changed to specify an `nml` folder under the station root absolute (^). If a destination folder does not already exist on the target host, the step creates it.

**NOTE:** The destination string must always begin with the character for either the system-home relative (!) or the station-home relative (^). The system provides no means to modify any files outside of the release directory on any target hosts.

When the job runs, the system makes a temporary copy of the file on the Supervisor unless the file being copied is local to the Supervisor (that is, you are using Workbench on the Supervisor). Once the job completes, the system deletes this temporary file.

To run the provisioning job more efficiently, the system combines file copy steps with other file copy steps, install software steps, or upgrade out-of-date software steps.

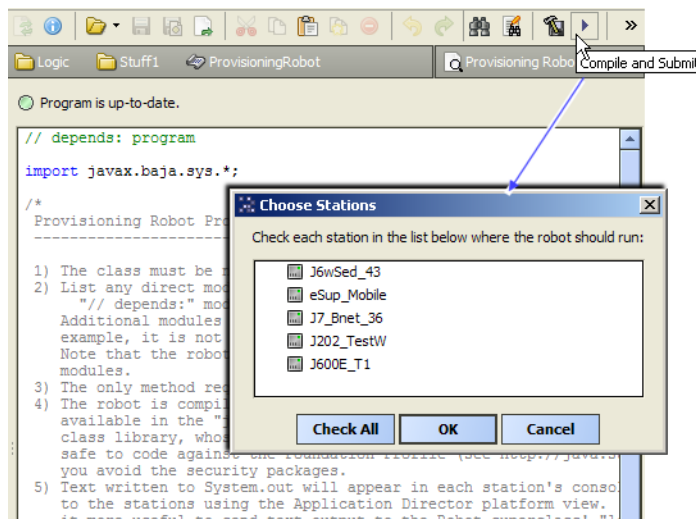
## Creating a provisioning robot

Using a provisioning robot you can customize a provisioning job to perform steps beyond the built-in steps provided. To successfully customize, you should already be familiar with the Baja class structures and methods, as well as the Java syntax used by Baja program code.

**Prerequisites:** You must have super user permissions to add or edit the Program and Robot components, including ProvisioningRobots. The `niagara.program.requireSuperUser` entry in the Supervisor's `system.properties` file controls this requirement.

- Step 1 Copy the `ProvisioningRobot` component from the `provisioningNiagara` palette into the Supervisor station.
- Step 2 Modify the robot's program code as appropriate.
- Step 3 Click the compile and save button (🔍).

The system opens the **Choose Stations** window.



- Step 4 Select one or more stations and click **OK**.

The **Niagara Network Job Builder** opens with the Run Robot step and selected stations entered, ready for you to add additional steps or launch the job..

## Adding a robot step to a provisioning job

The Robot step adds an existing robot to a provisioning job for running on each remote host.

**Prerequisites:** Super user permissions are not required to configure a provisioning job that includes a Run Robot step, where the referenced `ProvisioningRobot` was previously added or edited by a super user.

And, the station user in any remote host used by the Supervisor for client access (fox) must be a super user for such a provisioning job to run successfully on the station.

**Step 1** Drag a station's **ProvisioningRobot** into the step list area in the **Niagara Network Job Builder** or **Niagara Network Prototype View**.

This adds the Run Robot step.

# Chapter 4 Provisioning Job management

## Topics covered in this chapter

- ◆ Job execution
- ◆ Provisioning data files
- ◆ Histories related to provisioning

Provisioning uses batch jobs as the method for doing most of its tasks. A combination of objects, including components, files, and histories are used to model provisioning jobs, and their contained steps.

Batch jobs are different from other jobs run by the Supervisor's `JobService`, because:

- The file records of a batch job execution are retained until the job is explicitly disposed of, instead of being held temporarily as children of the `JobService`, then quickly deleted.
- Batch jobs can optionally trigger alarms (alerts) upon successful or unsuccessful completion.
- Batch jobs are first dispatched to a thread pool job queue, before being given to the `JobService`.
- Histories are automatically created for provisioning batch jobs as well as for individual device steps.

These sections describe those objects and their relationship to each other.

## Job execution

Just prior to execution, some steps in a provisioning batch job are combined. Currently, the only steps that are combined are Install Software, Copy File, and Upgrade Out-of-date Software steps that are adjacent to each other. Combining steps avoids duplication of dependency-checking with a station and minimizes the number of reboots required.

Upon execution, the provisioning batch job first executes any initial steps to run only once. **NiagaraNetwork** provisioning this means that the system runs the Update Licenses step (if included) first. Typically, the Supervisor has Internet connectivity, and makes a single, silent inquiry to the licensing server, passing the current licensing information for each host running an included station (in the job) to the licensing server. The licensing server responds with updated licensing information (if any) for these hosts in the form of a license archive, where any updated licenses are installed, as well as updated within the Supervisor's local license database. For more background details, refer to related Niagara 4 Platform Guide sections.

Following this, the provisioning batch job executes the remaining steps in sequence for each station, working through its list of stations in sequence. When the job reaches a station in the list, its station state reports `Running`. If any step fails, the station's state reports `Failed`, no additional steps are run for that station, and the job continues with the next station in the list.

If every step succeeds for a station, the station state reports `Success`. If the job is canceled during a station step, the station state and that of all following stations in the list report `Canceled`.

When all steps are complete for all stations without canceling, and all steps complete successfully, the job state reports `Success`. However, if even one step failed, the job state reports `Failed`.

## Provisioning data files

Provisioning creates a number of files on the Supervisor PC.

Two types of files back up provisioning jobs:

- Batch job log files and batch step log files are binary records of each completed provisioning batch job and step. The system writes these files under the Supervisor's **Station** directory using the following convention:

```
^batchJob/logs/batchJob_type/timestamp.bjl (or .bjsl)
```

- Station files are .dist files and software snapshots. The system writes these files under the station's **Files** node.

To see the station files, station users require admin-level Read (R) permissions on a category assigned to the **provisioningNiagara** subfolder.

Backup .dist files are written under the Supervisor's **Station** directory using the following convention:

```
^provisioningNiagara/stationData/station/backups/backup_station_timestamp.dist
```

- Snapshots of the software installed on the hosts running the stations are written under the Supervisor's **Station** directory using this convention:

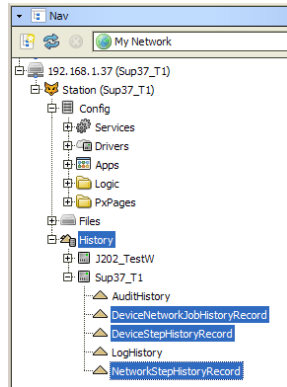
```
^provisioningNiagara/stationData/station/software/snapshot.bog
```

**CAUTION:** Do not manually delete any provisioning files from the Supervisor PC, for example, by using Windows Explorer, or in Workbench by using the **My File System** node in the Nav tree. Instead, use the **Dispose** button on various provisioning views. This button is on the **Niagara Network Job List** and **Prototype Job List** view. When you dispose of a job, the system automatically deletes all its related batch job log files and provisioning station data files.

## Histories related to provisioning

The system automatically records provisioning jobs within the history space of the Supervisor.

Figure 5 Provisioning related histories in Supervisor's history space



The system automatically creates these histories using the following names:

- **DeviceNetworkHistoryRecord** contains one record for each provisioning job attempted. The record includes various fields that record the job's finished state, submitting user, stations to process, and whether the job has been disposed of.
- **DeviceStepHistoryRecord** contains one record for each run for each station step, per station, for any provisioning job. The record includes the step's type, finished state, description, station, and whether the step was disposed of (that is, it was included in a job that was disposed of).
- **NetworkStepHistoryRecord** contains one record for each initial step to run only once (that is the Update Licenses step). The record includes the step's type, finished state, description, network type, and stations to process.

**CAUTION:** Do not rename, delete, or clear these histories on the Supervisor, for example by using the **Database Maintenance** view on the Supervisor's history space. These histories are used by the station's Batch-Job API code, and by various job list views.

Deleting these histories can result in provisioning-related files becoming orphaned, such that it becomes impossible to know if they can safely be deleted. Therefore, examination of these histories is optional, and in most cases you can simply ignore them.





# Chapter 5 Components

## Topics covered in this chapter

- ◆ Components in the batchJob module
- ◆ Components in the provisioningNiagara module

Component include services, folders and other model building blocks associated with a module. They may be dragged and dropped onto a Property or Wire sheet from a palette.

The descriptions included in the following topics appear as headings in documentation. 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**

Following is a list of the components in the **template** module:

## Components in the batchJob module

There are two components in the batchJob module: **BatchJobService** and **ThreadPoolJobQueue**.

### batchJob-BatchJobService

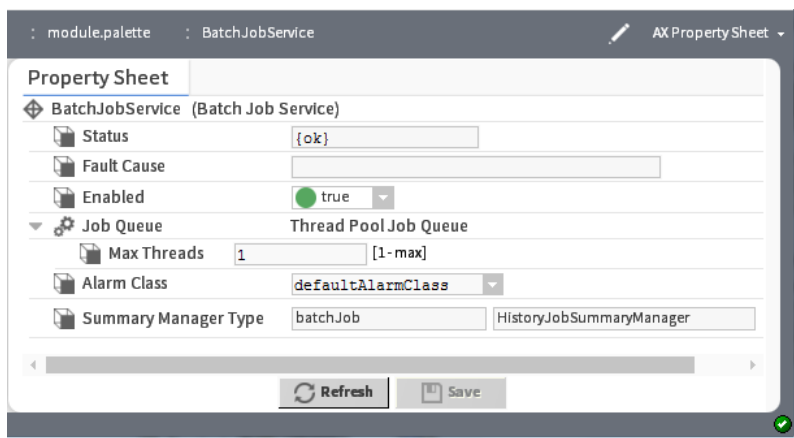
This service provides job control functions for provisioning jobs, including sending each provisioning job (as a batch job) to be run by the station's **JobService**.

The **BatchJobService** is required in the Supervisor station to facilitate provisioning. After copying it from the **batchJob** or **provisioningNiagara** palettes into the Supervisor's **Services** folder, you do not normally interface with it, apart from specifying the station's alarm class to use for provisioning alarms.

The **BatchJobService** requires the Supervisor station to also have the **HistoryService** and **JobService**, otherwise the **BatchJobService** is in *fault*. Typically, any Supervisor already has these services.

In addition, the **BatchJobService** requires the Supervisor host platform to be licensed with the provisioning feature, or else the service is in *fault*.

Figure 6 BatchJobService properties



Property	Value	Description
Status [component]	read-only text: ok disabled fault	Displays the current state of the component or extension.  The platform connection will be in <code>fault</code> if any of the following occurs: <ul style="list-style-type: none"> <li>Supervisor has no <code>ProvisioningNwExt</code> under its <code>NiagaraNetwork</code> (for example, it has been deleted).</li> <li>Supervisor is not licensed for provisioning.</li> <li><code>NiagaraStation</code> is in <code>fault</code>.</li> <li>The station's platform daemon rejects the platform connection's credentials.</li> </ul> The extension is <code>disabled</code> if its <code>Enabled</code> property is set to <code>false</code> or the <code>ProvisioningNwExt</code> is disabled.
Enabled	true (default) or false	If set to <code>false</code> , provisioning activity via the <code>BatchJobService</code> cannot occur.
Fault Cause	text	Read-only field. Indicates why the network, component, or extension is in fault.
Job Queue	thread pool job queue	Manages the submission of provisioning batch jobs by using a thread pool, to ensure that the Supervisor's CPU and network resources are not overtaxed by concurrent sessions. A single property, <code>Max Threads</code> , defaults to one (1), meaning only one provisioning job can run at a time. No special views or other features are provided.  Only after determining the station has available resource overhead, should <code>Max Threads</code> be adjusted to values over 2 or 3. Otherwise, other tasks performed by the station may be affected.
Alarm Class		The Supervisor station's <code>AlarmService</code> to be used for alarms (technically, alerts) when provisioning batch jobs fail and/or complete—as set using the alarm check boxes of the view used to build the provisioning job ( <code>Niagara Network Job Builder</code> and <code>Niagara Network Prototype View</code> ).

## batchJob-ThreadPoolJobQueue

This frozen container slot under the `BatchJobService` has only one property: .

`Max Threads` specifies the maximum number of concurrent provisioning jobs that can be performed by the Supervisor. By default, this is one (1), and is sometimes best left at default, as provisioning threads can be resource intensive in a Supervisor station.

**NOTE:** Only after determining that the station has available resource overhead, should `Max Threads` be adjusted up over 2 or 3. Otherwise, other tasks performed by the station may be affected.

## Components in the provisioningNiagara module

These components provide a variety of services.

Five component extensions are automatically added to every `NiagaraStation` under the Supervisor's `NiagaraNetwork`, providing the Supervisor also has the `ProvisioningNwExt` (`ProvisioningNiagaraNetworkExt`) under its `NiagaraNetwork`. The five provisioning extensions are:

- `BackupStationExt`


- `PlatformConnection`
- `SoftwareStationExt`
- `LicenseStationExt`
- `StationProxy`

The remaining components are:

- `BackupStationExt`
- `FileCopyStep`
- `InstallableSummary`
- `InstallableSpec`
- `InstallBySpecStep`
- `InstallStep`
- `LicenseStationExt`
- `PlatformConnection`
- `ProvisioningBackupStep`
- `NiagaraNetworkJob`
- `NiagaraNetworkJobPrototype`
- `ProvisioningNiagaraNetworkExt`
- `RebootJobStep`
- `SoftwareContainer`
- `SoftwareStationExt`
- `StationPollScheduler`
- `StationProxy`
- `UpgradeOutOfDateStep`

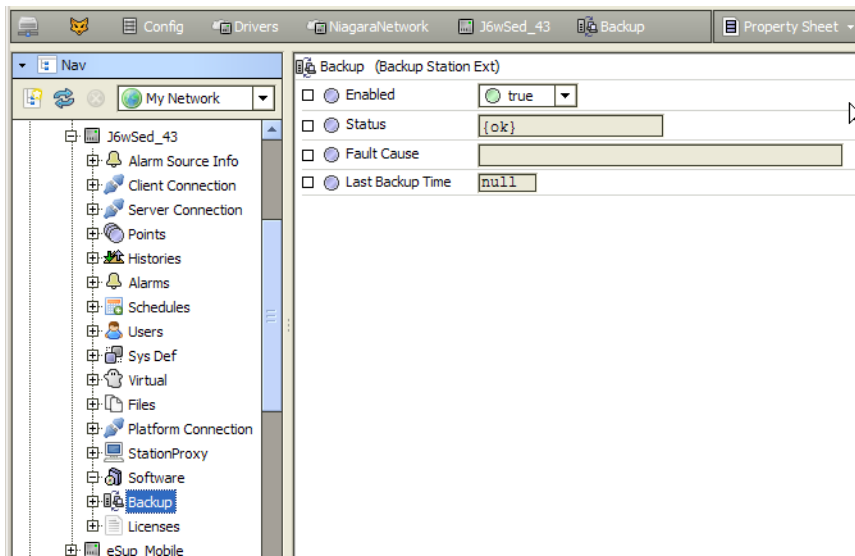
## provisioningNiagara-BackupStationExt

This component supports the Backup Stations step in provisioning jobs. It is one of the five device extensions that are automatically added to every station under the Supervisor's **NiagaraNetwork**. By default the extension is enabled.

This icon identifies the backup component: .

This component requires that the `ProvisioningNwExt` component be resident under its **NiagaraNetwork**. No special views exist for this extension.

Figure 7 Backup Station Ext property sheet



## Properties


Property	Value	Description
Enabled	true or false; defaults to true	To prevent the use of the Backup Station step in any provisioning job created for this station, change this value to false.
Status [component]	read-only text: ok disabled fault	Displays the current state of the component or extension.  The platform connection will be in <i>fault</i> if any of the following occurs: <ul style="list-style-type: none"> <li>Supervisor has no <b>ProvisioningNwExt</b> under its <b>NiagaraNetwork</b> (for example, it has been deleted).</li> <li>Supervisor is not licensed for provisioning.</li> <li><b>NiagaraStation</b> is in <i>fault</i>.</li> <li>The station's platform daemon rejects the platform connection's credentials.</li> </ul> The extension is <i>disabled</i> if its <b>Enabled</b> property is set to false or the <b>ProvisioningNwExt</b> is disabled.
Fault Cause	text	Read-only field. Indicates why the network, component, or extension is in fault.

## Start Backup action

When invoked, the Start Backup action immediately submits a provisioning job that contains a single Backup Stations step against this station. To schedule the backup to run at a specific time or at a specific interval for this particular station, you can link a **TriggerSchedule** output to this action. An even better practice is to add a job prototype component to the station and configure it to do perform the backup using its own **TriggerSchedule** to set the time. This method allows you to configure job retention, which provides automatic backup job disposition.

## provisioningNiagara-InstallableSummary

This component facilitates the creation of a list of software files under the `!sw` directory of the Supervisor host, and displays the files under the `SoftwareContainer` of the **NiagaraNetwork's ProvisioningNwExt**. This container contains all installed software modules, including modules added to the registry after the initial scan. To manage this container you use the **Supervisor Software Manager**.

This icon identifies the backup component: .

Each installable summary contains one or more specification objects for each version in the software registry. Each object is represented as an `InstallableSpec` component. At station startup, this service starts a thread that scans the software registry and populates the `SoftwareContainer`.


Direct children of the `SoftwareContainer` are summary objects (`InstallableSummary` components) for named, typed software files (for example, file type module named "baja"). For each summary object, there is a specification object (`InstallableSpec` components) for each version in the registry.

Apart from these summary children, the `SoftwareContainer` has but a single frozen property: `loaded`—a boolean slot that indicates if the startup thread has finished scanning the registry (by default, it is hidden).

Your key interface to the `SoftwareContainer` is its default view: the **Supervisor Software Manager**.

## provisioningNiagara-InstallableSpec


These components are children of `InstallableSummary` components. They occupy the lowest level under the `SoftwareContainer` in the **NiagaraNetwork's ProvisioningNwExt** component.

This icon identifies the backup component: .

`InstallableSpecs` are version-specific, and describe each software item that can be installed on a remote host, including version number, dependencies and other data.

## provisioningNiagara-InstallStep


This component supports the Install backupdist steps the system creates when you copy an existing backup `.dist` file into the middle job step pane (To run for each station) in either the **Niagara Network Job Builder** or the **Niagara Network Prototype View**.

This icon identifies the backup component: .

This step differs from the Install Software steps supported by `InstallBySpecStep`. It does not require selection by version number.

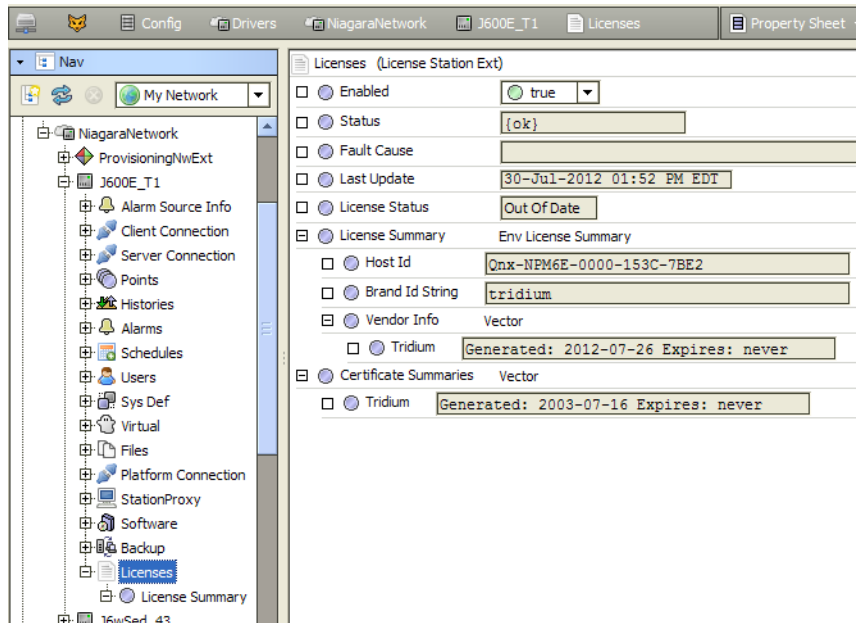
## provisioningNiagara-LicenseStationExt

This component supports the Update Licenses step, which updates the licenses on all remote hosts included in the job. `LicenseStationExt` is one five device extensions the system adds automatically to every NiagaraStation under the Supervisor's **NiagaraNetwork**. You access this view by double-clicking a license row in the **ProvisioningNwExt's Licenses** slot.

This icon identifies the backup component: .

By default this component is enabled. The properties you configure on its property sheet update the values reported in the **ProvisioningNwExt's Licenses** view. No special views exist for this extension.

Figure 8 License Station Ext property sheet



## Properties

Property	Value	Description
Enabled	true (default) or false	To prevent a provisioning job from running the Update Licenses step against this station, set this property to <i>false</i> .
Status [component]	read-only text: ok disabled fault	Displays the current state of the component or extension. The platform connection will be in <i>fault</i> if any of the following occurs: <ul style="list-style-type: none"> <li>Supervisor has no <b>ProvisioningNwExt</b> under its <b>NiagaraNetwork</b> (for example, it has been deleted).</li> <li>Supervisor is not licensed for provisioning.</li> <li><b>NiagaraStation</b> is in <i>fault</i>.</li> <li>The station's platform daemon rejects the platform connection's credentials.</li> </ul> The extension is <i>disabled</i> if its <b>Enabled</b> property is set to <i>false</i> or the <b>ProvisioningNwExt</b> is disabled.
Fault Cause	text	Read-only field. Indicates why the network, component, or extension is in <i>fault</i> .
Last Update	date and time	Reports the date and time that a provisioning job updated the license last. This property reports null if the license has not yet been updated this way.


Property	Value	Description
License Status	Up-to-Date, Unknown, Expired	Indicates the state of the license: <ul style="list-style-type: none"> <li>Up-to-Date reports that the license is the most recent available.</li> <li>Unknown indicates that provisioning has never updated the license in this station.</li> <li>License Summary provides a container slot to hold properties describing the license(s). The information held by this slot includes the unique Host ID for the platform, brand identifiers, and other vendor license information with timestamps to indicate when each license was originally generated, and its expiration date (if ever).</li> </ul>
Certificates Summaries	text	Provides a container slot to hold properties that describe installed certificate(s), including when each certificate was originally generated, and its expiration date (if ever).

### Poll action

Each Licenses station provisioning extension provides its own **Poll** action, which when invoked immediately submits an Update request to retrieve the latest license from the licensing server (or if unavailable, from the Supervisor’s local license database). This action is equivalent to the “**Update**” command issued from the **License Summary View** of the **ProvisioningNwExt’s Licenses** slot.

### provisioningNiagara-PlatformConnection

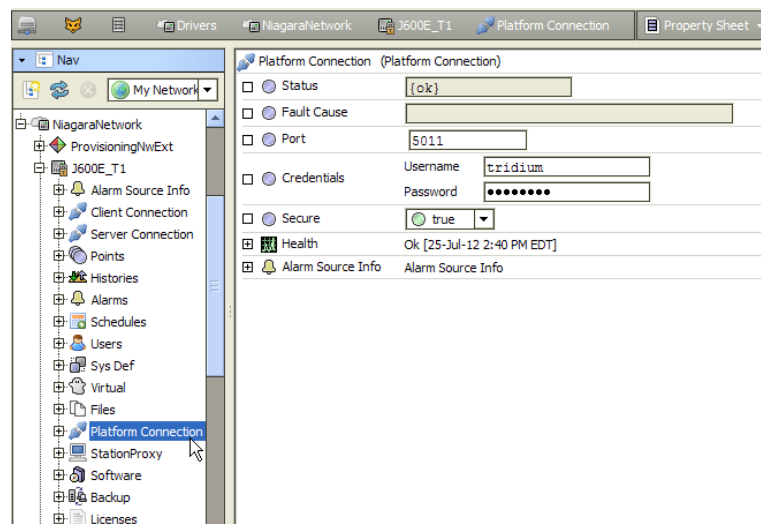
This component is used by the **ProvisioningNwExt** to poll each station.

This icon identifies the backup component: .

To configure each host’s newly-created PlatformConnection component, you access the Platform Connection property sheet by right-clicking **ProvisioningNwExt** and clicking **Views→Property Sheet**. Then you specify both the connection’s HTTP port and platform credentials (username and password). No special view, other than the property sheet, exists for this component.

Most provisioning jobs use secure platform connections (platformssl for versions 3.7 and 3.8; platformtls for versions 4.0 and later).

Figure 9 Platform Connection property sheet (AX-3.7 station shown)



## Properties

Property	Value	Description
Status [component]	read-only text: ok disabled fault	Displays the current state of the component or extension.  The platform connection will be in <code>fault</code> if any of the following occurs: <ul style="list-style-type: none"> <li>Supervisor has no <code>ProvisioningNwExt</code> under its <b>NiagaraNetwork</b> (for example, it has been deleted).</li> <li>Supervisor is not licensed for provisioning.</li> <li><b>NiagaraStation</b> is in <code>fault</code>.</li> <li>The station's platform daemon rejects the platform connection's credentials.</li> </ul> The extension is <code>disabled</code> if its <code>Enabled</code> property is set to <code>false</code> or the <code>ProvisioningNwExt</code> is disabled.
Fault Cause	text	Read-only field. Indicates why the network, component, or extension is in fault.
Port	number (defaults to 3011 for a connection that is not secure; defaults to 5011 for a secure connection)	Identifies the port on which the platform daemon in the station's host is listening.  Displays as <code>PlatformPort</code> in the <b>Add</b> or <b>Edit</b> windows when working in the <b>Station Manager</b> view.
Credentials	Username and password	Specifies the credentials used for a platform connection to the host's running station. Credentials display as <code>PlatformUser</code> and <code>PlatformPassword</code> in the <b>Add</b> or <b>Edit</b> windows when working in the <b>Station Manager</b> view.
Secure Platform	<code>true</code> (must be <code>true</code> for N4.0 and later) or <code>false</code> (default for AX-3.7 and AX-3.8)	Specifies if a secure connection should be used to the host. Displays as <code>SecurePlatform</code> in the <b>Add</b> or <b>Edit</b> dialog in the <b>Station Manager</b> view.
Health	text	Contains information about the success or failure of the last pings, and is similar to the standard "Health" slot in most driver networks.
Alarm Source Info		Specifies how and if alarms are to be generated as a result of ping monitor failures, similar to the standard <b>Alarm Source Info</b> slot in most driver networks.

## Action

A single `Ping` action is available on the `PlatformConnection`, to immediately force a short message to the host's platform daemon. Its `Health` property updates with ping results. To test their validity, issue this action after entering port and credentials properties.

## provisioningNiagara-SoftwareStationExt

This component supports the steps that a provisioning job can run on a station. By default this component is always enabled. Other functions provided by the station are available using the **Station Software View**.


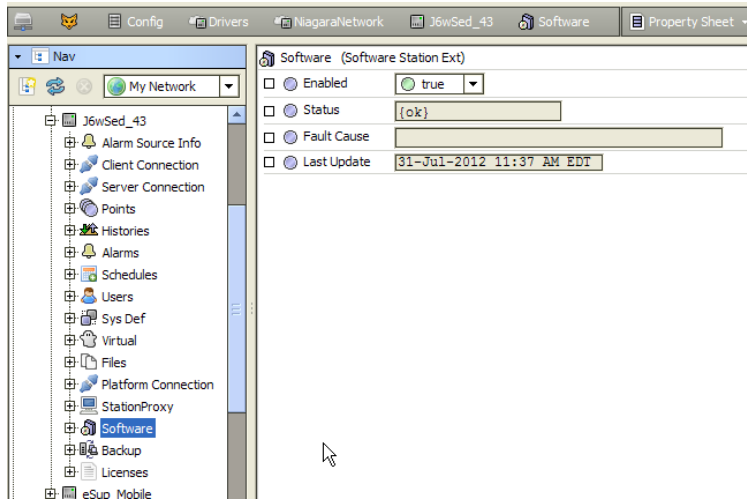
This icon identifies the backup component: .




Figure 10 Software Station Ext property sheet



Property	Value	Description
Enabled	defaults to <code>true</code>	Changing this property to <code>false</code> prevents the processing of any steps against this station. <b>Enabled</b> must be set to true to use the <b>Station Software View</b> and <b>Supervisor Software Manager</b> on this station.
Status [component]	read-only text: <code>ok</code> <code>disabled</code> <code>fault</code>	Displays the current state of the component or extension. The platform connection will be in <code>fault</code> if any of the following occurs: <ul style="list-style-type: none"> <li>Supervisor has no <b>ProvisioningNwExt</b> under its <b>NiagaraNetwork</b> (for example, it has been deleted).</li> <li>Supervisor is not licensed for provisioning.</li> <li><b>NiagaraStation</b> is in <code>fault</code>.</li> <li>The station's platform daemon rejects the platform connection's credentials.</li> </ul> The extension is <code>disabled</code> if its <b>Enabled</b> property is set to <code>false</code> or the <b>ProvisioningNwExt</b> is disabled.
Fault Cause	text	Read-only field. Indicates why the network, component, or extension is in fault.
Last Update	date and time	Documents the last platform snapshot. This property is null if a platform snapshot has never occurred.

### provisioningNiagara-StationProxy

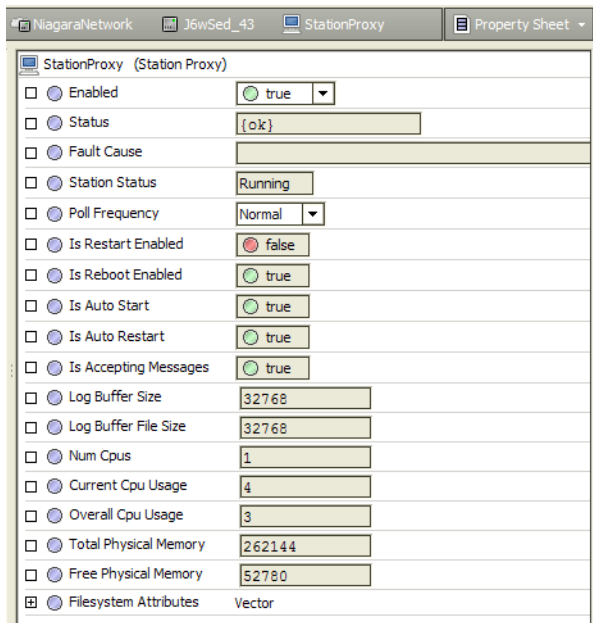
This component provides platform administration functions like those available when you open a direct platform connection in Workbench, using the **Station Director** and **Platform Administration** views. It also provides a number of actions for station control functions.

This icon identifies the backup component: .

**StationProxy** is one of five device extensions the system automatically adds to every **NiagaraStation** under the Supervisor's **NiagaraNetwork**. By default, this component is enabled to allow polling from the **ProvisioningNwExt** for values of the extension's properties.

This component’s default view is the **Provisioning Station Director**.

Figure 11 StationProxy property sheet



The **StationProxy** component provides other special views, including the default view (**Provisioning Station Director**) as well as a **Station Job List**.

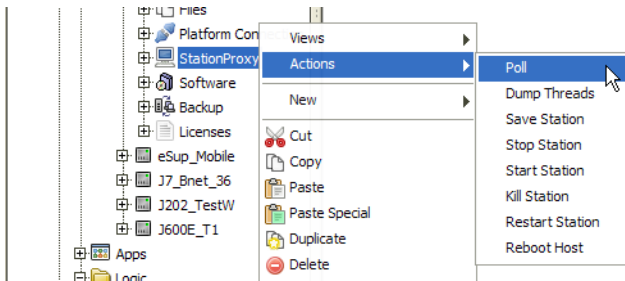
**Properties**

Property	Value	Description
Enabled	true (default) or false	To prevent polling by the <b>ProvisioningNwExt</b> , set this property to <b>false</b> . To use the special views on the <b>Station Proxy</b> extension, namely the <b>Provisioning Station Director</b> and <b>Station Job List</b> , <b>Enabled</b> must be <b>true</b> .
Status [component]	read-only text: ok disabled fault	Displays the current state of the component or extension. The platform connection will be in <b>fault</b> if any of the following occurs: <ul style="list-style-type: none"> <li>Supervisor has no <b>ProvisioningNwExt</b> under its <b>NiagaraNetwork</b> (for example, it has been deleted).</li> <li>Supervisor is not licensed for provisioning.</li> <li><b>NiagaraStation</b> is in <b>fault</b>.</li> <li>The station’s platform daemon rejects the platform connection’s credentials.</li> </ul> The extension is <b>disabled</b> if its <b>Enabled</b> property is set to <b>false</b> or the <b>ProvisioningNwExt</b> is disabled.
Fault Cause	text	Read-only field. Indicates why the network, component, or extension is in fault.
Status [station]	text	Reflects one of the following values: <ul style="list-style-type: none"> <li>Idle — Station is not currently running, and can be started without a reboot.</li> </ul>

Property	Value	Description
		<ul style="list-style-type: none"> <li>Starting — Station process is running, but has not completed its startup sequence.</li> <li>Running — Station is running.</li> <li>Stopping — Station is in process of shutting down, but its process is still alive.</li> <li>Halted — Station is not currently running, and the host must be rebooted before it can start.</li> <li>Unknown — <code>StationProxy</code> status is <code>disabled</code> or <code>fault</code>. As a result, station status is unknown. Status is also unknown if the station is unreachable, or if a poll has not happened yet.</li> </ul>
Poll Frequency		Corresponds to the <b>Poll Scheduler</b> in the <code>ProvisioningNwExt</code> , as part of its monitor ping mechanism (a ping of the platform daemon in the host running each station).
Is Restart Enabled	read-only	If <code>true</code> , the station you can restart the station without a reboot of its host platform (such as with Win32-based platforms).
Is Reboot Enabled	read-only	If <code>true</code> , the host's platform daemon is capable of (and allows) rebooting of the host.
Is Auto Start	read-only	If <code>true</code> , the station restarts automatically after the host reboots.
Is Auto Restart	read-only	If a station terminates with a failure exit code and this property is <code>true</code> , the host restarts (or reboots) if <code>Is Restart Enabled = false</code> .
Is Accepting Messages	read-only	If <code>false</code> (unlikely), thread dumps, station saves, and graceful shutdown are not possible using the platform daemon.
Log Buffer Size	read-only	The size (in bytes) of the buffer used by the platform daemon to hold the console output.
Log Buffer File Size	read-only	The maximum size of the <code>console.txt</code> file (in bytes) that the platform daemon saves console output to when the station stops.
Num Cpus	read-only	The number of CPUs on the host running the station.
Current Cpu Usage	read-only	The percentage of time the CPU(s) have been in use in the last second.
Overall Cpu Usage	read-only	The percentage of time the CPU(s) have been in use since the platform daemon started.
Total Physical Memory	read-only	The total KB of physical RAM on the station's host.
Free Physical Memory	read-only	The KB of available physical RAM on the station's host.
File System Attributes	read-only	Free space statistics for each file system on station's host.

## Actions

Figure 12 Action menu for StationProxy extension



Many of these actions are also available in the Provisioning Station Director view, as well as in views using a direct platform connection. When invoked, each action performs as follows:

Action	Description
Poll	Causes Supervisor to poll the host's platform daemon for current data.
Dump Threads	Supervisor requests that the station send a thread dump to its console output.
Save Station	Supervisor requests that the station save its current state to its own (local) config.bog file. requests that the station save its current state to its own (local) config.bog file.
Stop Station	Supervisor requests that the station shuts down gracefully.
Start Station	Supervisor requests the platform daemon to start the station. This action is applicable only if current station status is idle.
Kill Station	Supervisor requests for the station to terminate immediately, without graceful shutdown.
Restart Station	Depending on <b>Is Restart Enabled</b> value, causes one of the following: <ul style="list-style-type: none"> <li>If <b>Is Restart Enabled</b> is <code>false</code> the station's host is rebooted.</li> <li>If <b>Is Restart Enabled</b> is <code>true</code> the station is stopped gracefully, then restarted again.</li> </ul>
Reboot Host	Depending on <b>Is Reboot Enabled</b> value, causes one of the following: <ul style="list-style-type: none"> <li>If <b>Is Reboot Enabled</b> is <code>false</code> nothing happens.</li> <li>If <b>Is Reboot Enabled</b> is <code>true</code>, the Supervisor requests for the platform daemon to shut down gracefully, then reboot the host.</li> </ul>

# Chapter 6 Plugins (views)

## Topics covered in this chapter

- ◆ Plugins in batchJob module
- ◆ Plugins in provisioningNiagara module

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, select **Help→On View (F1)** from the menu or press **F1** while the view is open.

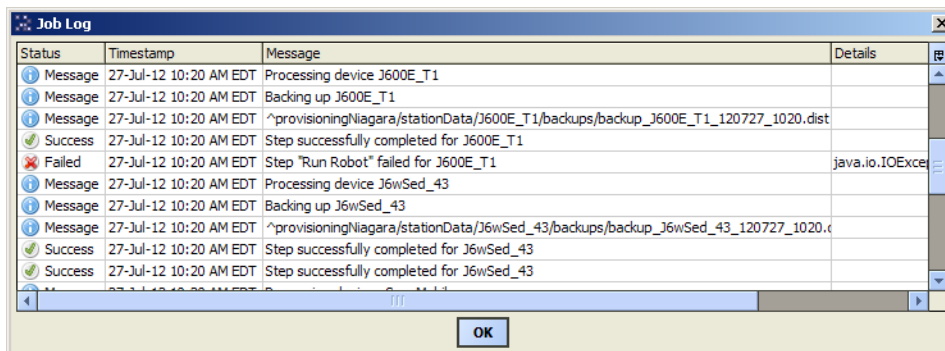
## Plugins in batchJob module

The provisioning batchJob module views list jobs and steps.

### Job Log

The popup **Job Log** table opens a running log of messages output with the execution of the job. Each row includes a column for status, timestamp, message, and details.

Figure 13 Example of a Job Log

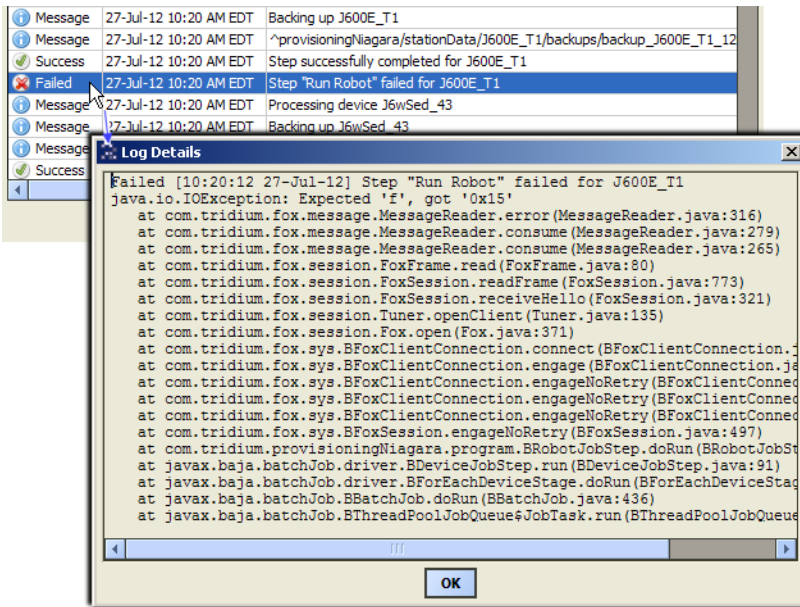


Status	Timestamp	Message	Details
Message	27-Jul-12 10:20 AM EDT	Processing device J600E_T1	
Message	27-Jul-12 10:20 AM EDT	Backing up J600E_T1	
Message	27-Jul-12 10:20 AM EDT	^provisioningNiagara/stationData/J600E_T1/backups/backup_J600E_T1_120727_1020.dist	
Success	27-Jul-12 10:20 AM EDT	Step successfully completed for J600E_T1	
Failed	27-Jul-12 10:20 AM EDT	Step "Run Robot" failed for J600E_T1	java.io.IOException
Message	27-Jul-12 10:20 AM EDT	Processing device J6wSed_43	
Message	27-Jul-12 10:20 AM EDT	Backing up J6wSed_43	
Message	27-Jul-12 10:20 AM EDT	^provisioningNiagara/stationData/J6wSed_43/backups/backup_J6wSed_43_120727_1020.c	
Success	27-Jul-12 10:20 AM EDT	Step successfully completed for J6wSed_43	
Success	27-Jul-12 10:20 AM EDT	Step successfully completed for J6wSed_43	

**NOTE:** New log messages do not appear dynamically—to see newer messages you must reopen the **Job Log**.

To see more details (if available) on any row, double-click it for a **Job Details** popup dialog, as shown below.

Figure 14 Log Details example from Job Log

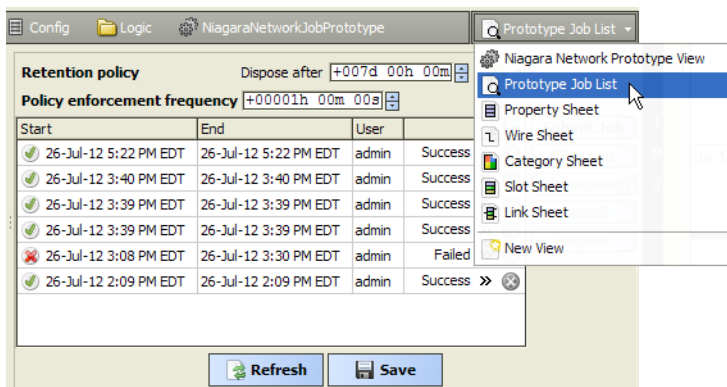


Column	Value	Description
Status		
Timestamp		
Message		
Details		

### batchJob-PrototypeJobList

This view provides a table-based history of this batch job and its results. It is available on any reusable provisioning job component (**NiagaraNetworkJobPrototype**), and is accessed by using the view selector, or by right-clicking the component and selecting **Views**→**Prototype Job List**.

Figure 15 Prototype Job List is another view of a NiagaraNetworkJobPrototype component

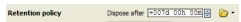
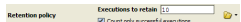


This view has just one list area, a table with buttons on the right and the bottom. It differs from the **Niagara Network Job List** (**NiagaraNetwork's ProvisioningNext** view) in the following ways:

- The two retention properties for the associated job prototype component are shown at top, where you can adjust if needed. Setting retention policy is recommended for any job that includes ongoing periodic backups.
- The table shows only jobs from the (one) associated job prototype (provisioning job), whereas the **Niagara Network Job List** table shows all retained provisioning jobs.

## Retention Policy

These rules determine what happens to the completed provisioning job.

Retention policy rule	Value	Description
Retain permanently	n/a	Causes all executed batch jobs to remain in the Supervisor's station job management system until manually disposed of.
Dispose after a specified amount of time	days, hours, and minutes (default 7 days) 	Causes executed batch jobs to be deleted after a period of time relative to the job's end timestamp.
Keep a limited number of executions	number and check box  The check box defaults to count only successful executions.	Causes the system to save only the most recent number of executed batch jobs, after which older jobs are deleted. Removing the selection causes the system to include all job attempts.
Policy enforcement frequency	hours, minutes, seconds (default is one hour)	Establishes the periodic frequency at which the job's retention policy is evaluated and enforced. You can manually invoke the <b>Enforce Retention Policy</b> action on the job component.

## Columns

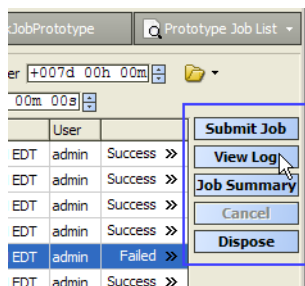
This main area of the **Prototype Jobs List** shows provisioning jobs that have been sent to run, are running, or are have completed. Any pending jobs do not appear until the job prototype's linked trigger schedule actually fires.

Column	Value	Description
Start or Started [job]	date and time (read-only)	Displays the date and time that the system submitted the job to the job queue.
End or Ended [job]	date and time	The date and time when the job stopped running. This property is blank if the job is still running.

Column	Value	Description
User [provisioning]	text	The station user that requested the job. This column displays <b>unknown</b> if job was triggered by a linked schedule.
Status or State [of the job]	read-only text:	<p>The current or final state of the job, as one of the following. The first three states appear on the <b>Device Network Job</b> view.</p> <ul style="list-style-type: none"> <li>Unknown —the job is pending execution.</li> <li>Running — the job is executing.</li> <li>Canceling — request to cancel the job was sent, but has not been processed yet, and the job is still executing.</li> <li>Success — job finished successfully, with all steps completed for all stations.</li> <li>Canceled — job was canceled before it completed, and is no longer running.</li> <li>Failed — at least one step failed in one station; job is no longer running.</li> </ul> <p>Each row in the table ends with a details button (&gt;&gt;) and a dispose button (X) . Clicking this button changes the view to the <b>Niagara Network Job</b> view or the <b>Batch Job Step Log File</b> view, which shows all logged messages that are related to this single job.</p> <p>The overall status for the job in other stations, may be different).</p>

### Buttons

Figure 16 Buttons along right side of Prototype Job List view.



Button	Value	Description
Submit Job	always enabled	Lets you request that this provisioning job run now. The system adds a new job row at the top of the jobs table, dynamically updating the job status to indicate the job’s progress.
View Log	button enabled when a job row is selected	Opens a popup <b>Job Log</b> window that displays the messages output by the selected job or step.
Job Summary	button	Changes to the <b>Niagara Network Job View</b> for the purposes of displaying detailed information about the selected job.



Button	Value	Description
Cancel or Cancel Job	button enabled only if a job is running	Clicking this button notifies the system to cancel the job when it is safe to do so. Not all job steps can be canceled.
Dispose	button enabled when a job is finished	<p>Clicking this button prompts you to confirm that you want to delete the selected job(s). The deletion includes all associated job files. If you confirm the deletion:</p> <ol style="list-style-type: none"> <li>1. The system deletes the job from the <b>JobService</b> (if it is still there and not rolled off as the 11th job, or station restart)</li> <li>2. The system deletes all associated job files including the batch job log file, batch log step log files, and other files if applicable. For a backup job, this includes deleting the backup .dist file(s).</li> <li>3. The system removes the job from the <b>Jobs Table</b> in the <b>ProvisioningNwExt's Niagara Network Job List</b>.</li> </ol> <p>You can select multiple jobs to dispose of at the same time.</p>

## Plugins in provisioningNiagara module

Provisioning Niagara includes use of the following `provisioningNiagara` views (listed alphabetically).

- Backup Step Details View
- Network License Summary
- Niagara Network Job Builder
- Niagara Network Job List
- Niagara Network Job View
- Niagara Network Prototype View
- Provisioning Manager
- ProvisioningRobotEditor
- Provisioning Station Director
- Station Job List
- Station Software View
- Supervisor License Manager
- Supervisor Software Manager

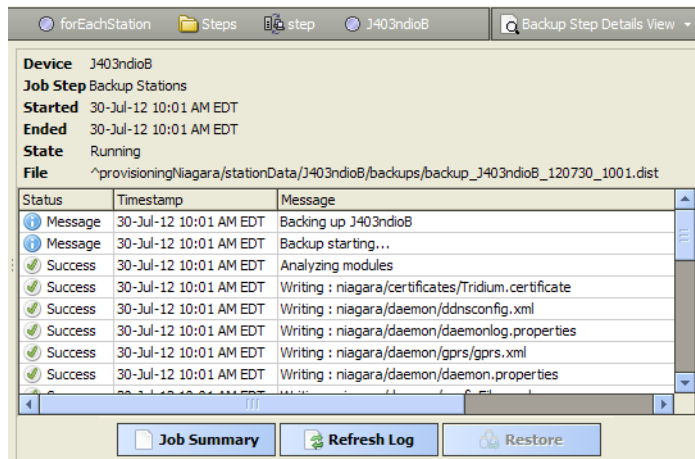
### provisioningNiagara-BackupStepDetailsView

This view shows the details for a Backup Stations step executed against a single station, while that step is still running. You access this from the Niagara Network Job View or the Station Job List by clicking ">>" next to the running step.

If the job step is other than a Backup Stations step, and the step is still running, you see a slightly different view, the **Niagara Network Job View**. In either case, once the step finishes, neither view is accessible. Both are replaced by the **Batch Job Step Log File View**.

This view provides a summary table of log messages that occur during step execution. To view additional **Log Details**, if any, double-click an individual row.

Figure 17 Backup Step Details View with details on one running backup step for a station



## Step elements

Elements, columns and buttons	Value	Description
Device	element	Identifies the station that is being processed or has been processed.
Job Step	element	Identifies the type of provisioning step: Backup Stations, File Copy, Install Software, etc.
Start or Started [step]	date and time	Displays the date and time that the step began processing.
End or Ended [job]	date and time	The date and time when the job stopped running. This property is blank if the job is still running.
Status or State [of the step]	read-only text	<p>The current or final state of each step:</p> <ul style="list-style-type: none"> <li>Running — the step is executing.</li> <li>Canceling — the request to cancel the step was sent, but has not been processed yet. The step is still running.</li> <li>Success — the step finished successfully.</li> <li>Canceled — the step was canceled before it completed and is no longer running.</li> <li>Failed — the step did not complete</li> </ul> <p>Each row in the table ends with a details button (&gt;&gt;) and a dispose button (X). This button functions the same as the <b>Step Details</b> button at the bottom of the view.</p> <p>The overall status for the step in other stations, may be different).</p>
File	text (visible only if a Backup Stations step)	Identifies the file path and name on the Supervisor for the saved backup .dist file. It uses the convention: ^provisioningNiagara/stationData/stationName/backups/backup_stationName_yymmdd_hhmm.dist

## Columns

Columns	Value	Description
Status or State [of the step]	read-only text	<p>The current or final state of each step:</p> <ul style="list-style-type: none"> <li>Running — the step is executing.</li> <li>Canceling — the request to cancel the step was sent, but has not been processed yet. The step is still running.</li> <li>Success — the step finished successfully.</li> <li>Canceled — the step was canceled before it completed and is no longer running.</li> <li>Failed — the step did not complete</li> </ul> <p>Each row in the table ends with a details button (&gt;&gt;) and a dispose button (X) . This button functions the same as the <b>Step Details</b> button at the bottom of the view.</p> <p>The overall status for the step in other stations, may be different).</p>
Timestamp	date and time	Displays the date and time when the log message was written.
Message	text	The actual log message.

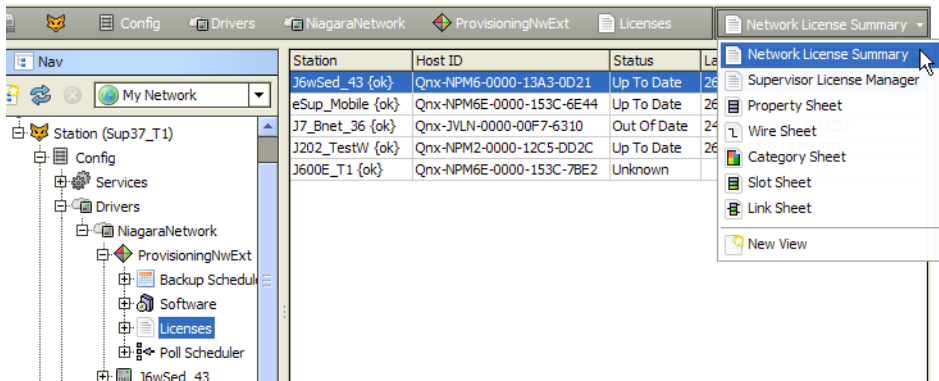
## Buttons

Button	Value	Description
Job Summary	button always enabled	Changes to the <b>Batch Job Log File View</b> for the job that contains this step.
Refresh Log	button enabled when information needs to be refreshed	Recreates the log.
Restore	button available if the job step is Backup Station and the backup completed successfully	Restores the station using the .dist file saved from this provisioning job. If you answer <b>Yes</b> to the confirmation window (no undo), an install backup job executes immediately and the view changes to the <b>Niagara Network Job View</b> .

## Network License Summary

This view provides a summary table listing the currently known license information for each station (**NiagaraStation**) in the network. It is the default view for the **SupervisorLicenses** slot on the **ProvisioningNext** under the Supervisor's **NiagaraNetwork**.

Figure 18 Network Licenses Summary



Each row contains the license information for a host running a station. The `SupervisorLicenses` device extension of each child station populates the table. If you double-click on a row, the view changes to the `SupervisorLicenses` extension property sheet for that particular `NiagaraStation`.

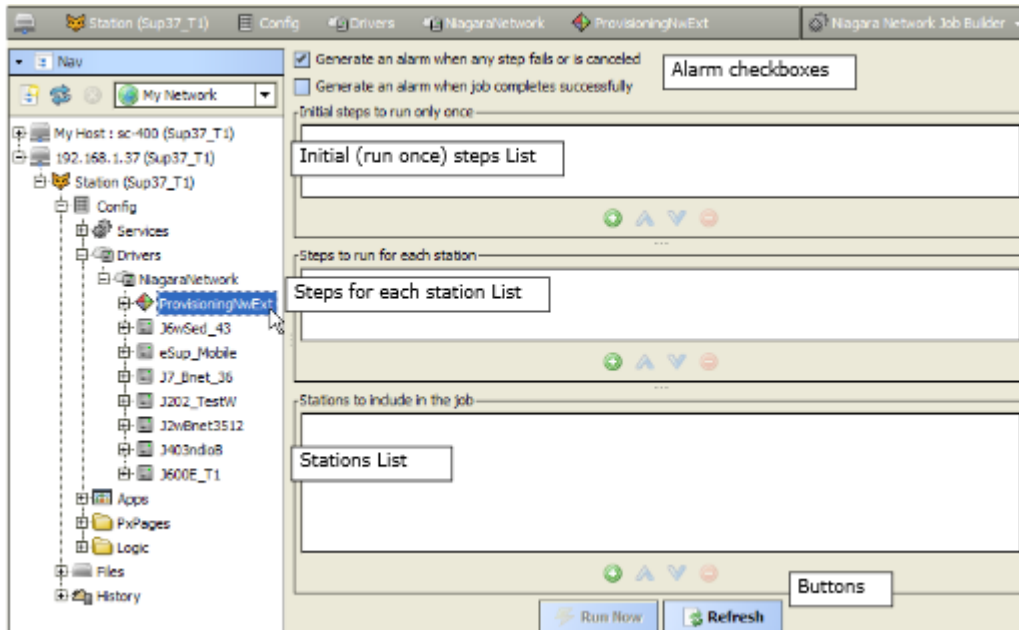
Column	Value	Description
Station	text	Identifies the name of the station.
Host ID	text	A 20-character identifier that provides unique identification for each host.
Status	Up-To-Date	A status of <code>Up To Date</code> means that the license on the remote host agrees with the license that the Supervisor has for it in its (own) local license database. It may be possible that a more recent license is available for it on the licensing server.
Last Updated	date and time	The timestamp when the station's license was last updated.

### provisioningNiagara-NiagaraNetworkJobBuilder

This view allows you to create a one-time provisioning job for immediate execution (to run now). It is the default view on the `ProvisioningNwExt` component. To access this view, double-click the `ProvisioningNwExt` component in the Nav tree, or right-click the component node and select **Views**→**Niagara Network Job Builder**.

**NOTE:** To build a job that you can schedule or run at a later time, save, duplicate and modify, use a `NiagaraNetworkJobPrototype` component copied anywhere in the Supervisor's station (each has its own equivalent view). You can find them in the `provisioningNiagara` palette. In general, those are the components you should use to create regularly scheduled station backup jobs.

Figure 19 Niagara Network Job Builder is the default view of ProvisioningNwExt



Using this view, you specify the steps to be performed by the one-time job, and for which stations. Then you submit the job without saving the job as a reusable component





**Alarm check boxes**

Alarm check box	Value	Description
Generate an alarm when any step fails or is canceled	check box	Both alarm check boxes determine if alarms (actually <i>alerts</i> ) are to be issued by the <b>BatchJobService</b> for this provisioning job, and under what circumstances. Alarms use the alarm class specified in the property sheet of the <b>BatchJobService</b> . They appear in the alarm console as alerts.  When selected, the <b>BatchJobService</b> raises an alarm whenever a job step fails or is canceled.
Generate an alarm when job completes successfully	check box	When selected, the <b>BatchJobService</b> raises an alarm whenever a job completes with no step failures.

## Steps and stations

List area	Value	Description
Initial steps to run only once	text	Provides a one-line summary for each step to be performed once in the job, regardless of how many stations the job specifies. Currently, when provisioning Niagara this means only one type of step: <b>Update Licenses</b> , and is optional.
Steps to run for each station	text	Provides a one-line summary for each step to be run for each station specified in this job. In most provisioning jobs, you add one or more steps.
Stations to include in the job	text	Lists all the stations to be processed by the job. This means each station processes all steps in the (middle pane) <b>Steps for Each Station List</b> . Only stations in the Supervisor's <b>Niagara Network</b> can be added. For any job, you add one or more stations, and you can also remove and reorder stations (stations are processed in a top-to-bottom order).

## Controls

Control	Description
	Adds a step.
	Removes the selected step.
	Moves the selected step up in the sequence of steps.
	Moves the selected step down in the sequence of steps.

## Buttons

Button	Value	Description
Run Now	button enabled when there is at least one job step in either the <b>Initial Steps To Run Once List</b> or <b>Steps for Each Station List</b> , and one station in the <b>Stations List</b> .	Clicking this button dispatches the job to the batch job queue for immediate execution. The Workbench view automatically changes to the <b>Niagara Network Job</b> view.
Refresh	button always enabled	Removes all entries from all three lists.

## Niagara Network Job view

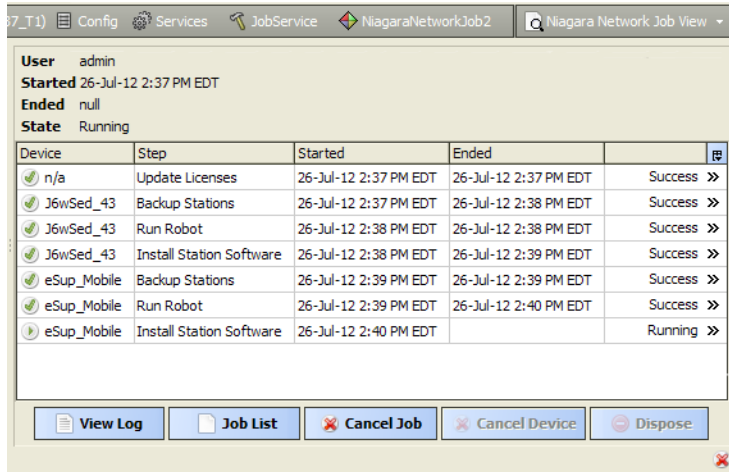
This view shows the details for the execution of a single provisioning job and is the default view for any **NiagaraNetworkJob** component.

You can access this view several ways:

- By clicking **Run Now** in the **Niagara Network Job Builder** or **Niagara Network Prototype View**.

- By clicking the “>>” button in the **Niagara Network Job List** or **Prototype Job List** on any provisioning job that is still running.  
If the job has completed and no longer appears in the **Job Service Manager** view, the system displays the **Batch Job Log File** view instead of this view. The **Batch Job Log File** functions the same as the **Niagara Network Job View**.
- By clicking **Job Summary** on the **Batch Job Step Log File View**.

Figure 20 Niagara Network Job View lists steps in one job



From top to bottom, this view has three areas:

- Job elements, which are read-only.
- A summary table of the steps in the job.
- A series of buttons at the bottom.

**Job elements**

Element	Value	Description
User [provisioning]	text	The station user that requested the job. This column displays unknown if job was triggered by a linked schedule.
Start or Started [job]	date and time (read-only)	Displays the date and time that the system submitted the job to the job queue.

Element	Value	Description
End or Ended [job]	date and time	The date and time when the job stopped running. This property is blank if the job is still running.
Status or State [of the job]	read-only text:	<p>The current or final state of the job, as one of the following. The first three states appear on the <b>Device Network Job</b> view.</p> <ul style="list-style-type: none"> <li>• <b>Unknown</b> —the job is pending execution.</li> <li>• <b>Running</b> — the job is executing.</li> <li>• <b>Canceling</b> — request to cancel the job was sent, but has not been processed yet, and the job is still executing.</li> <li>• <b>Success</b> — job finished successfully, with all steps completed for all stations.</li> <li>• <b>Canceled</b> — job was canceled before it completed, and is no longer running.</li> <li>• <b>Failed</b> — at least one step failed in one station; job is no longer running.</li> </ul> <p>Each row in the table ends with a details button (&gt;&gt;) and a dispose button (X) . Clicking this button changes the view to the <b>Niagara Network Job</b> view or the <b>Batch Job Step Log File</b> view, which shows all logged messages that are related to this single job.</p> <p>The overall status for the job in other stations, may be different).</p>

## Columns

Column	Value	Description
Device	element	Identifies the station that is being processed or has been processed.
Step	column	Identifies the type of step.
Start or Started [step]	date and time	Displays the date and time that the step began processing.



Column	Value	Description
End or Ended [step]	date and time	The date and time when the step stopped running. This property is blank if the job is still running.
Status or State [of the step]	read-only text	<p>The current or final state of each step:</p> <ul style="list-style-type: none"> <li>Running — the step is executing.</li> <li>Canceling — the request to cancel the step was sent, but has not been processed yet. The step is still running.</li> <li>Success — the step finished successfully.</li> <li>Canceled — the step was canceled before it completed and is no longer running.</li> <li>Failed — the step did not complete</li> </ul> <p>Each row in the table ends with a details button (&gt;&gt;) and a dispose button (X) . This button functions the same as the <b>Step Details</b> button at the bottom of the view.</p> <p>The overall status for the step in other stations, may be different).</p>

## Buttons

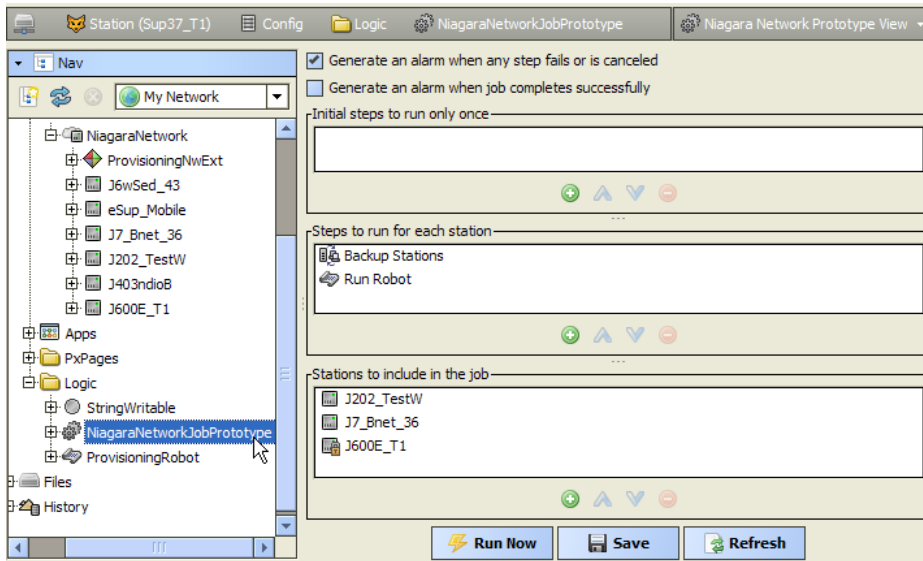
Button	Value	Description
View Log	button enabled when a job row is selected	Opens a popup <b>Job Log</b> window that displays the messages output by the selected job or step.
Job List	button always enabled	Clicking this button opens the <b>Niagara Network Job List</b> .
Cancel or Cancel Job	button enabled only if a job is running	Clicking this button notifies the system to cancel the job when it is safe to do so. Not all job steps can be canceled.
Cancel Device	button enabled only if a job is running and a step row is selected	For the selected station only, clicking this button notifies the system to cancel the job when it is safe to do so. The system begins processing the job for the next station.
Dispose	button enabled when a job is finished	<p>Clicking this button prompts you to confirm that you want to delete the selected job(s). The deletion includes all associated job files. If you confirm the deletion:</p> <ol style="list-style-type: none"> <li>The system deletes the job from the <b>JobService</b> (if it is still there and not rolled off as the 11th job, or station restart)</li> <li>The system deletes all associated job files including the batch job log file, batch log step log files, and other files if applicable. For a backup job, this includes deleting the backup .dist file(s).</li> <li>The system removes the job from the <b>Jobs Table</b> in the <b>ProvisioningNwExt's Niagara Network Job List</b>.</li> </ol> <p>You can select multiple jobs to dispose of at the same time.</p>

## provisioningNiagara-NiagaraNetworkPrototypeView

The **Niagara Network Prototype View**, which is nearly identical to the **ProvisioningNwExt's Niagara Network Job Builder**, is the default view on each reusable job prototype component (**NiagaraNetworkJobPrototype**). To access this view, double-click the **NiagaraNetworkJobPrototype** component, or right-click the component node and select **Views→Niagara Network Prototype View**.

You use this view to specify and edit a specific provisioning job.

Figure 21 Niagara Network Prototype View



As on the **Niagara Network Job Builder** view, this view has alarm check boxes, three list areas with controls, and buttons at the bottom of the view.





### Alarm check boxes

Alarm check box	Value	Description
Generate an alarm when any step fails or is canceled	check box	Both alarm check boxes determine if alarms (actually <i>alerts</i> ) are to be issued by the <b>BatchJobService</b> for this provisioning job, and under what circumstances. Alarms use the alarm class specified in the property sheet of the <b>BatchJobService</b> . They appear in the alarm console as alerts.  When selected, the <b>BatchJobService</b> raises an alarm whenever a job step fails or is canceled.
Generate an alarm when job completes successfully	check box	When selected, the <b>BatchJobService</b> raises an alarm whenever a job completes with no step failures.

## Steps and stations

List area	Value	Description
Initial steps to run only once	text	Provides a one-line summary for each step to be performed once in the job, regardless of how many stations the job specifies. Currently, when provisioning Niagara this means only one type of step: <b>Update Licenses</b> , and is optional.
Steps to run for each station	text	Provides a one-line summary for each step to be run for each station specified in this job. In most provisioning jobs, you add one or more steps.
Stations to include in the job	text	Lists all the stations to be processed by the job. This means each station processes all steps in the (middle pane) <b>Steps for Each Station List</b> . Only stations in the Supervisor's <b>Niagara Network</b> can be added. For any job, you add one or more stations, and you can also remove and reorder stations (stations are processed in a top-to-bottom order).

## Controls

Control	Description
	Adds a step.
	Removes the selected step.
	Moves the selected step up in the sequence of steps.
	Moves the selected step down in the sequence of steps.

## Buttons

Button	Value	Description
Run Now	button enabled when there is at least one job step in either the <b>Initial Steps To Run Once List</b> or <b>Steps for Each Station List</b> , and one station in the <b>Stations List</b> .	Clicking this button dispatches the job to the batch job queue for immediate execution. The Workbench view automatically changes to the <b>Niagara Network Job</b> view.  If unsaved changes exist and the <b>Save</b> button enabled, a pop-up window asks if you wish to save the configuration before queuing it to run.
Save	button enabled when unsaved changes exist.	Click this button to immediately save the changes to the corresponding job prototype component.
Refresh	button always enabled	Removes all entries from all three lists.

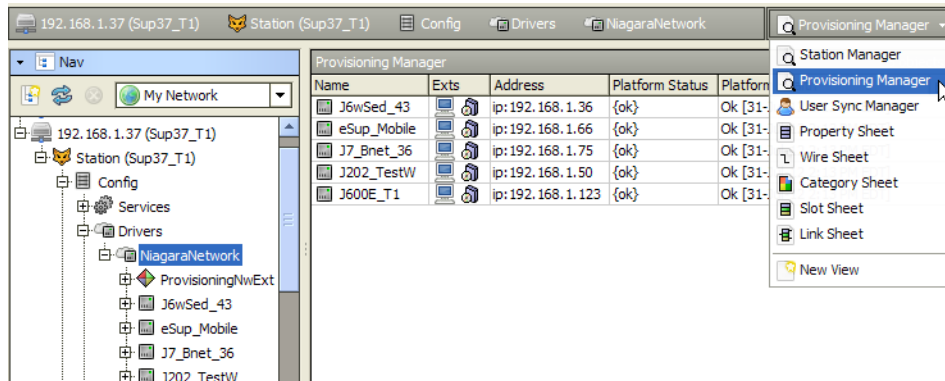
## provisioningNiagara-ProvisioningManager

This view provides a central look at the status and health of platform connectivity to the various remote hosts, as well as quick access to some of the provisioning (device) extensions under each **NiagaraStation**. It

is an available view on the Supervisor's **NiagaraNetwork**, provided that the Supervisor is licensed for provisioning and has the **ProvisioningNwExt** installed.

The view is based on a table, where each row represents a **NiagaraStation** component (similar to the network's default **Station Manager** view).

Figure 22 Provisioning Manager view of the Supervisor's NiagaraNetwork



## Columns

Column	Value	Description
Name	text	The identifying name of the station.
Exts	n/a	Provides double-click access to two provisioning device extensions views: <ul style="list-style-type: none"> <li><b>Provisioning Station Director</b> for <b>StationProxy</b></li> <li><b>Station Software Manager</b> for <b>Software</b></li> </ul> To access the other provisioning extensions, expand a <b>NiagaraStation</b> in the Nav tree or use the property sheet of a <b>NiagaraStation</b> .
Address	IP format	The IP address of the station.
Platform Status		Displays the current condition of the provisioning extension.
Platform Health		This information is updated by the ongoing ping monitor to the platform daemon. For any station (row), you can also right-click it and manually issue a <b>Ping</b> action.

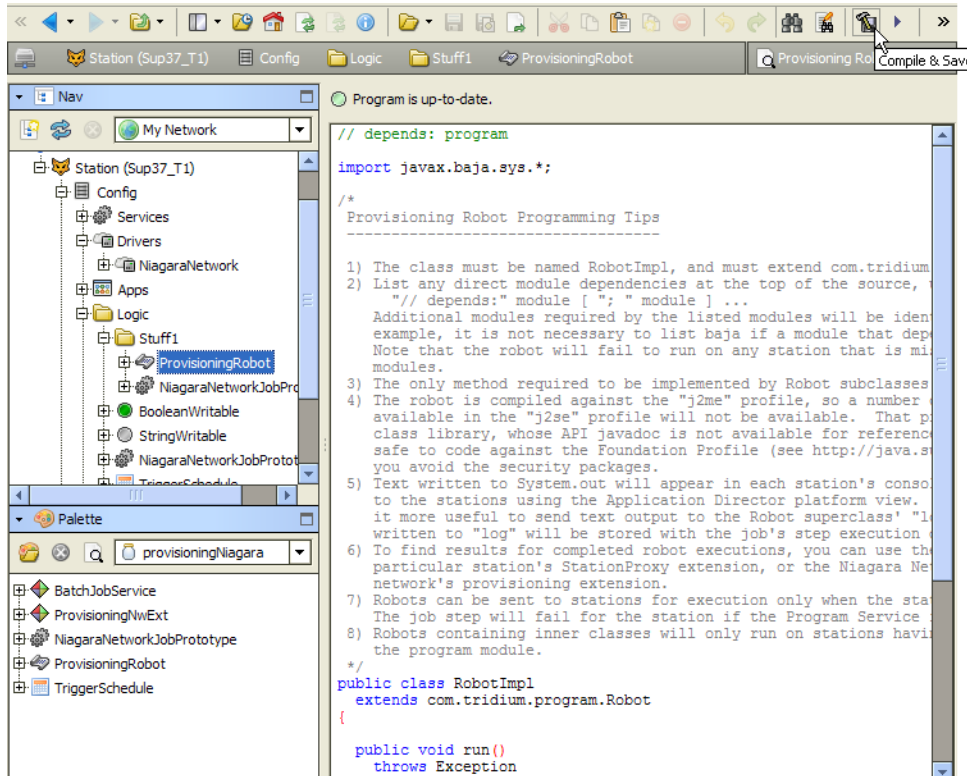
## Buttons

Button	Value	Description
Edit	enabled when one or more rows is selected.	Provides a way to modify the platform connection credentials and port. These credentials provide provisioning access to the Supervisor station. These credentials may also be modified using each <b>NiagaraStation's Platform Connection</b> (device extension) property sheet

## provisioningNiagara-ProvisioningRobotEditor

This view provides a program editing window that closely resembles the **Edit** tab of the **Program Editor** view for **Program** components, in that, you view, edit and compile the Baja code represented as a **ProvisioningRobot**. This view is the main view of the **ProvisioningRobot** component.

Figure 23 Provisioning Robot Editor is default view for a ProvisioningRobot



This view operates like the **Robot Editor** view of a station's **ProgramService**.

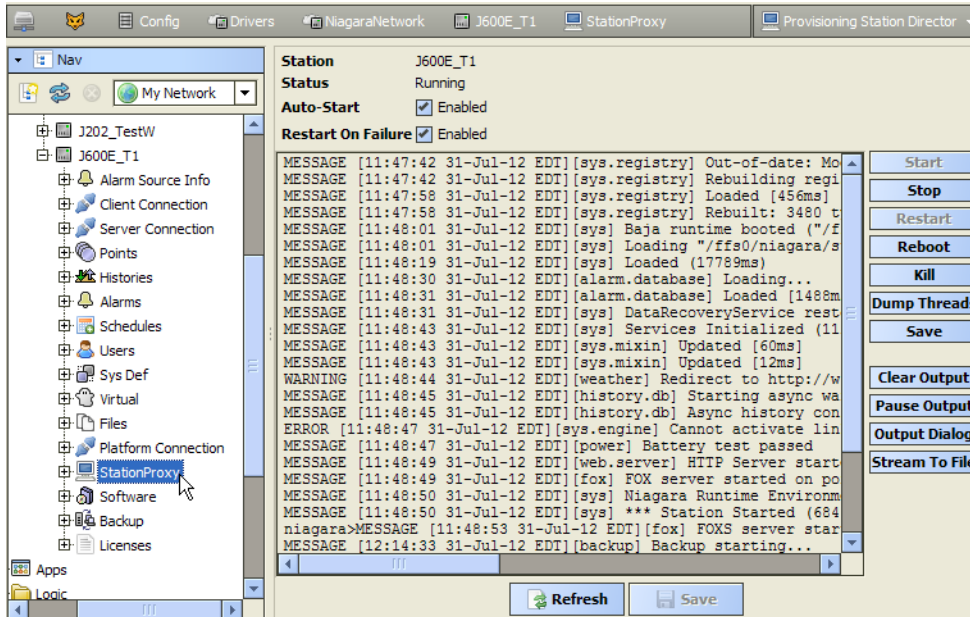
Button	Icon	Description
Compile and save		Compiles and saves the code.
Run now		Opens the <b>Choose Stations</b> window.

## provisioningNiagara-ProvisioningStationDirector

This view is the default view on the **StationProxy** provisioning extension of a **NiagaraStation**. This view closely resembles the **Station Director** view in a direct platform connection.

This view closely resembles the **Application Director** view available in a direct platform connection to a host.

Figure 24 Provisioning Station Director is default view on StationProxy extension



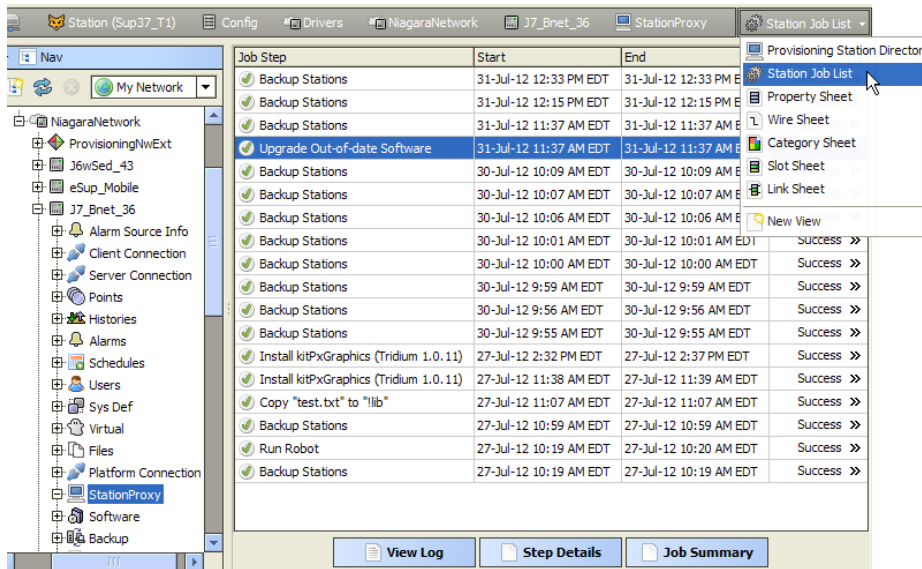
Refer to the *Application Director* section in the *Drivers Guide* for descriptions of most elements in this view. Only elements that differ from that view are explained here.

- Since the **Provisioning Station Director** only shows information for one station, it does not show the station name and status within a table (at the top of the view), but instead shows this data at the top using simple text labels.
- Where the **Application Director** updates the **Auto-Start** and **Restart on Failure** settings immediately when changed, the **Provisioning Station Director** works more like a normal view, meaning you must click the **Save** button after making any changes.
- Although the appearances of the two views are similar, their implementations are different. The **Application Director** connects the Workbench view directly to the station's platform daemon, and is best for extended troubleshooting. Whereas, the **Provisioning Station Director** uses the Supervisor station as an intermediary, and as a result is not as responsive, and is less efficient (uses additional Supervisor resources).

## provisioningNiagara-StationJobList

This view summarizes provisioning job steps that have been executed against this particular station, with additional details available. It is the default view of the **StationProxy** extension under a **NiagaraStation** device. You access this view using the view selector or by right-clicking the **StationProxy** extension and selecting **Views** → **Station Job List** or by selecting it from the extension's view selector.

Figure 25 Station Job List is available view on StationProxy extension



### Step table

This main area of the **Station Job List** view shows a row for each step that has been executed against the station. No record is available for a step’s execution unless it has started. For this reason, the following steps do not appear in this view:

- steps for jobs not yet started.
- steps for jobs that are running, but are still running prior steps.
- steps that come after any earlier steps (for any station) that were canceled.
- steps that would have executed after another step, but the other step failed for this station.

Because of this, the **Station Job List** is not the appropriate view to use to find the answer for questions like, “Why did the backup scheduled for Tuesday on this station not run?” For this type of information, look in the **Niagara Network Job List** of the **ProvisioningNwExt**.

The step table includes columns for various data. You can do any of the following within the table:

- To view any step’s **Step Log File View**, which is the same as using the **Step Details** button at the bottom of the view, click the “>>” (Details) button to the right of the status
- To view the **Job Log** for any job, double-click any step row. The Job Log lists a series of messages about the step. This is the same as using the **View Log** button at the bottom of the view.
- Right-click a step for a popup menu, providing the same functions as those provided by the buttons at bottom of view.

Column	Value	Description
Job Step	text	Identifies the type of job step, such as <b>Backup Stations</b> and so on.
Start or Started [job]	date and time (read-only)	Displays the date and time that the system submitted the job to the job queue.
End or Ended [job]	date and time	The date and time when the job stopped running. This property is blank if the job is still running.

Column	Value	Description
Status or State [of the job]	read-only text:	<p>The current or final state of the job, as one of the following. The first three states appear on the <b>Device Network Job</b> view.</p> <ul style="list-style-type: none"> <li>• <b>Unknown</b> —the job is pending execution.</li> <li>• <b>Running</b> — the job is executing.</li> <li>• <b>Canceling</b> — request to cancel the job was sent, but has not been processed yet, and the job is still executing.</li> <li>• <b>Success</b> — job finished successfully, with all steps completed for all stations.</li> <li>• <b>Canceled</b> — job was canceled before it completed, and is no longer running.</li> <li>• <b>Failed</b> — at least one step failed in one station; job is no longer running.</li> </ul> <p>Each row in the table ends with a details button (&gt;&gt;) and a dispose button (X) . Clicking this button changes the view to the <b>Niagara Network Job</b> view or the <b>Batch Job Step Log File</b> view, which shows all logged messages that are related to this single job.</p> <p>The overall status for the job in other stations, may be different).</p>

## Buttons

Button	Value	Description
View Log	button enabled when a job row is selected	Opens a popup <b>Job Log</b> window that displays the messages output by the selected job or step.
Step Detail	button always available	Switches the view to the <b>Batch Job Step Log File View</b> .
Job Summary	button always enabled	Changes to the <b>Batch Job Log File View</b> for the job that contains this step.

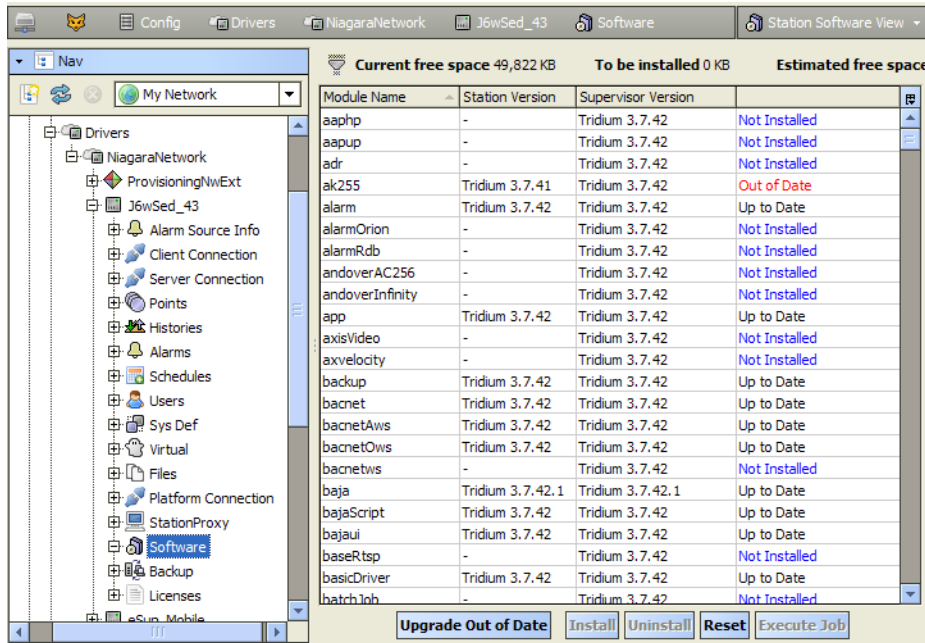
## provisioningNiagara-StationSoftwareView

This displays the current state of the station's software modules. It is the default view for the **Software** provisioning extension.

The **Station Software Manager** closely resembles the **Software Manager** that is available with a direct platform connection to a host. For more information about this view, refer to the *Software Manager* section in the *Platform Guide*. Only elements that differ from that view are explained here.



Figure 26 Station Software View is default view on Software provisioning extension



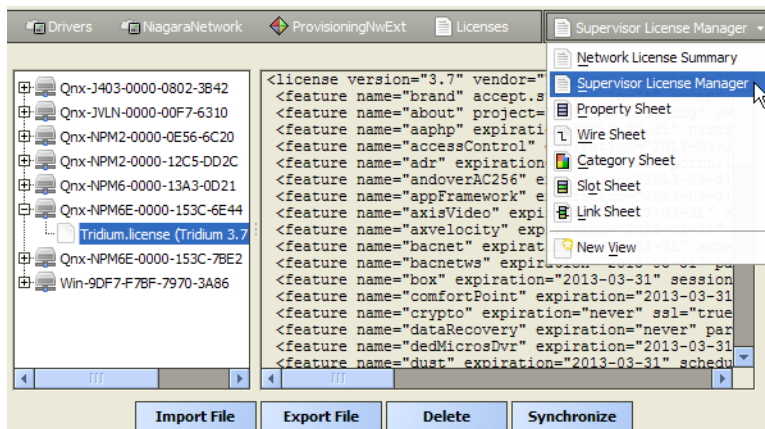
When you access this view, the system takes a snapshot of the station’s current software configuration and displays it as a table. Other differences from the **Software Manager** view in a direct platform connection are summarized as follows:

- The **Software Manager** provides columns for Installed Version and Available Version. These identify the version of each module installed on the station and available in Workbench. The **Station Software View** has equivalent columns labeled Station Version and Supervisor Version.
- Instead of a **Commit** button that starts the software installation by running it in Workbench, the **Station Software View** has an **Execute Job** button. To submit the installation as a provisioning batch job in the Supervisor station, you click **Execute Job**, which opens the **Niagara Network Job View**.

### provisioningNiagara-SupervisorLicenseManager

This view provides management access to the Supervisor’s local license database, which is located under its !licenses/db subdirectory. The **Supervisor License Manager** is an available view on the **Licenses** slot of the **ProvisioningNwExt** under the **NiagaraNetwork**. To access this view, double-click the **Licenses** slot.

Figure 27 Supervisor License Manager view is available view on ProvisioningNwExt’s License slot



The Supervisor’s local license database is the structured organization of “host ID-named” sub-folders under the Supervisor’s `!/licenses` folder that contain license files. As in the equivalent **Workbench License Manager** view, this view provides a two-pane window into all the license files and parent host ID folders, where the:

- Left pane provides tree navigation, where you can expand folders and click (to select) license files.
- Right pane shows the text contents of any selected license file.

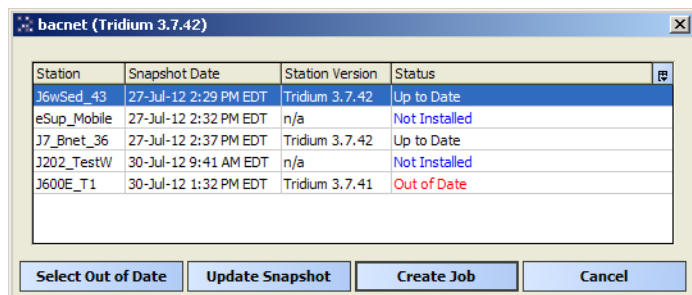
Buttons at the bottom of this view provide a way to manage the contents of the Supervisor’s local license database.

Button	Value	Description
Import File	always enabled	Adds license file(s) from a local licence file or license archive (.lar) file.
Export File	always enabled	Allows you to save all licenses (or any selected licenses) locally as a license archive file.
Delete	enabled when a row is selected	Allows you to remove licenses from the Supervisor’s local license database.
Synchronize	enabled if you have Internet connectivity	Allows you to update all licenses (or any selected licenses) in the Supervisor’s local license database with the most current versions on the online licensing server.  For more details and related information, refer to the Niagara 4 Platform Guide.

### Check stations list

Clicking the **Check Stations** button at the bottom of the **Supervisor Software Manager** window opens the check stations list.

Figure 28 Example window for installable from Check Stations function



The window shows one row for each **NiagaraStation**. The last column in the table shows the status of the platform snapshot for each station.

### Columns

Columns and buttons	Value	Description
Station	text	The station identifier.
Snapshot Date	date	The date the most recent state of the station was captured.

Columns and buttons	Value	Description
Station version	numbers	The version of the software in the platform snapshot (viewable in its <b>Software</b> extension).
Status	text	<p><b>Up to Date</b> indicates that the station version is equal to or greater than the software file in the equivalent file in the Supervisor station</p> <p><b>Out of Date</b> indicates that the version of the file in the Supervisor station is greater than the version installed on the station's host.</p> <p><b>Not Installed</b> indicates that this file is not installed on the station's host.</p> <p><b>No Snapshot</b> indicates that no platform snapshot has been taken for this station. There is no basis for comparison. Click to select the station row and click <b>Update Snapshot</b>.</p> <p><b>Bad Remote file</b> indicates that the version of the module on the station's host is corrupt or otherwise unusable.</p>

### Buttons

Buttons	Value	Description
Select Out of Date	always enabled	Causes the system to display the information for only stations whose software is not current.
Update Snapshot	enabled if a row is selected	Captures the current state of the software modules and stores it in the Supervisor database
Create Job	always enabled	Creates a new provisioning job.
Cancel	always enabled	Closes the window.



# Chapter 7 Troubleshooting

## Topics covered in this chapter

- ◆ Why the Start Backup action is NOT recommended

Use these notes to resolve provisioning problems.

**The Supervisor station's ProvisioningNwExt (under its NiagaraNetwork) and all provisioning-related extensions for NiagaraStations under its NiagaraNetwork have a fault status.**

A license for the provisioningNiagara feature is missing or expired. Only a Supervisor can be (or needs to be) licensed for provisioningNiagara.

**I attempted to drag the BatchJobService folder to my station Config container and got the message, "The following missing modules are required for root targets to be transefered: 'batchJob'."**

Your Supervisor station is not licensed for provisioning..

**When comparing the module databases on my Supervisor with those on each host station one of the station module files appears to be corrupted.**

Back up the station data, commission the station again, and restore the backup.

**I clicked the Hyperlink button on the Alarm Console in an effort to view the details of a provisioning alert and got the message, "Cannot Display Page." What is going on?**

You or someone else disposed of the related provisioning job before it was acknowledged. Disposing of a job removes the batch job log (.bll) file and all batch job step log (.bisl) files associated with the provisioning job. Provisioning alarms should be acknowledged before disposing of them.

## Why the Start Backup action is NOT recommended

The `Start Backup` action works well for systems with only a few controllers. As controllers multiply, there is a better way to back up all stations.

In a large system with many controllers connected to a single Supervisor, invoking the action to backup all stations (or equivalent steps listed) creates a provisioning job that can take an excessive amount of time to complete and can put an undue load on the system, especially if an administrator invokes it at a peak time. Finally, unlike a provisioning job from a **NiagaraNetworkJobPrototype**, which is the preferred backup method, the accumulated backup .dist files remain stored on the Supervisor until manually deleted. These files are cannot controlled to job retention policies. Without manual intervention, over a long period of time this could lead to a disk-full condition on a Supervisor.

For a large enterprise system where a Supervisor has many controllers, backing up is better accomplished by adding multiple **NiagaraNetworkJobPrototype** components in the Supervisor's station. You copy the components from the **provisioningNiagara** palette. Then you can configure each one for a custom backup job, selecting some of the system's host stations in each component.

To run each backup job at some periodic interval, perhaps at an off-hours time, you could add and link to a standard **TriggerSchedule** component (also available on the **provisioningNiagara** palette). By using multiple **TriggerSchedules** (one configured slightly differently for each linked **NiagaraNetworkJobPrototype**). Backups could constructively be staged in sequence—say 10 minutes apart from one another.

Additionally, each **NiagaraNetworkJobPrototype** component has configurable job retention policies, via its **Prototype Job List** view. You can (and should) configure them to provide automatic disposal of older saved backup .dist files, based on age or some number of earlier saved backup .dist files. For an example procedure including **NiagaraNetworkJobPrototypes**, see [Prototype jobs, page 22](#).



# Glossary

batch job	A station job that is managed (sent to the station's job service) by the station's BatchJobService. Provisioning functions using batch jobs.
job prototype	Refers to the <b>NiagaraNetworkJobPrototype</b> component found in the <b>provisioningNiagara</b> palette. This persisted component defines a provisioning job by containing one or more processing steps to be performed on a given list of host stations. To set up the regular performance of a task, such as backing up multiple stations, you drag this container to the wire sheet and connect the out slot ( <b>Trigger</b> slot) of a standard <b>TriggerSchedule</b> to the component's <b>Submit Job</b> action.
job	The mechanism used to manage a task that a station performs. Jobs run asynchronously in the background while providing user visibility regarding what is going on in the station. See also, provisioning job.
platform snapshot	A list of installed software on a remote host that is running a station. When you access the Software extension under a <b>NiagaraStation</b> component, the system builds or updates this list. Provisioning uses this list when performing queries and installing software.
provisioning	<p>Automating the tasks required to maintain host controllers in a station's NiagaraNetwork. For the most part, these are platform tasks—that is, they would otherwise be done using Workbench and an individual platform connection directly to a remote host.</p> <p>Using a provisioning robot you can run custom program code in the station running on each host. The stations' ProgramServices run the custom code. The Supervisor station performs all these tasks, modeled in the Supervisor station as provisioning jobs.</p> <p>Outside of provisioning, you must perform similar tasks manually using full Workbench.</p>
provisioning job	A sequence of steps to perform on one or more host stations. The steps provide a way to perform the same task on multiple stations, such as update licenses, back up, install software, copy files, etc.
software registry	The catalog of available software files, such as modules (.jars) or distribution files (.dists), that are located under the Supervisor's !sw directory and can be installed in a remote host.