

Technical Document

Niagara^{AX} Networking and IT Guide

October 9, 2006



Networking and IT Guide

Copyright © 2006 Tridium, Inc.
All rights reserved.
3951 Westerre Pkwy., Suite 350
Richmond
Virginia
23233
U.S.A.

Copyright Notice

The software described herein is furnished under a license agreement and may be used only in accordance with the terms of the agreement.

This document may not, 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.

The confidential information contained in this document is provided solely for use by Tridium employees, licensees, and system owners; and is not to be released to, or reproduced for, anyone else; neither is it to be used for reproduction of this Control System or any of its components.

All rights to revise designs described herein are reserved. While every effort has been made to assure the accuracy of this document, Tridium shall not be held responsible for damages, including 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 release and technology contained herein may be protected by one or more U.S. patents, foreign patents, or pending applications.

Trademark Notices

BACnet and ASHRAE are registered trademarks of American Society of Heating, Refrigerating and Air-Conditioning Engineers. Microsoft and Windows are registered trademarks, and Windows NT, Windows 2000, Windows XP Professional, and Internet Explorer are trademarks of Microsoft Corporation. Java and other Java-based names are trademarks of Sun Microsystems Inc. and refer to Sun's family of Java-branded technologies. 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, Niagara^{AX} and Vykon are registered trademarks, and Workbench, WorkPlace^{AX}, and ^{AX}Supervisor, 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. The software described herein is furnished under a license agreement and may be used only in accordance with the terms of the agreement.

CONTENTS

Preface contents

| | |
|-----------------------------|-----|
| Document change log | iii |
| About this document | iii |
| Related documentation | iv |

NiagaraAX introduction

| | |
|---|-----|
| NiagaraAX architecture | 1-2 |
| Types of NiagaraAX programs | 1-2 |
| Types of NiagaraAX protocols | 1-3 |
| <i>About the Fox protocol</i> | 1-3 |
| Types of NiagaraAX platforms | 1-4 |
| <i>Types of JACE controllers</i> | 1-5 |
| <i>NiagaraAX Supervisor</i> | 1-6 |
| <i>WorkPlaceAX (and the NiagaraAX platform)</i> | 1-6 |
| <i>Browser user interface</i> | 1-6 |

NiagaraAX and networking

| | |
|--|------|
| About IP addressing | 2-1 |
| Private IP addresses | 2-1 |
| Network address translation (NAT) | 2-2 |
| IP routing and default gateway | 2-2 |
| Routing example | 2-3 |
| Static and dynamic IP addressing in NiagaraAX | 2-4 |
| NiagaraAX and DHCP | 2-5 |
| About the HOSTS file | 2-6 |
| About DNS | 2-6 |
| Domain names | 2-6 |
| Name Servers | 2-6 |
| NiagaraAX and DNS | 2-7 |
| About WINS | 2-7 |
| About DDNS | 2-7 |
| About the NiagaraAX network | 2-8 |
| About non-NiagaraAX networks | 2-8 |
| Protocols | 2-9 |
| Types of NiagaraAX hosts | 2-10 |
| QNX-based NiagaraAX hosts | 2-10 |
| Win32-based NiagaraAX hosts | 2-10 |
| NiagaraAX host networking technologies | 2-11 |
| NiagaraAX network examples | 2-12 |

| | |
|---|-------------|
| NiagaraAX single-site network application | 2-12 |
| NiagaraAX multi-site network application | 2-14 |
| Managing NiagaraAX hosts | 2-15 |
| AXSupervisor | 2-16 |
| NiagaraAX platform | 2-16 |
| <i>About the platform connection</i> | 2-17 |
| <i>Types of platform administration functions</i> | 2-17 |
| <i>About the platform daemon on a PC</i> | 2-18 |
| Other host administration tools | 2-19 |
| NiagaraAX backups | 2-20 |
| NiagaraAX hosts and network communication | 2-21 |

NiagaraAX security

| | |
|--|-------------|
| NiagaraAX security model | 3-1 |
| NiagaraAX and Niagara R2 security comparison | 3-4 |
| NiagaraAX security considerations | 3-4 |
| General Security Guidelines | 3-5 |
| Guidelines for QNX-based NiagaraAX hosts | 3-5 |
| Guidelines for Win32-based NiagaraAX hosts | 3-5 |
| Creating a strong password | 3-6 |
| Firewalls and proxy servers | 3-7 |
| About virtual private networks | 3-7 |
| Example NiagaraAX VPN network | 3-8 |
| About NiagaraAX ports | 3-11 |

Network traffic considerations

| | |
|--|-------------|
| NiagaraAX network communications considerations | 4-2 |
| Types of browser communications | 4-3 |
| About the wbapplet | 4-3 |
| Workbench-to-station communications | 4-5 |
| Station-to-station communications | 4-7 |
| Fox service | 4-9 |
| Network tuning policies | 4-10 |
| <i>Tuning Policy properties</i> | 4-11 |
| Network polling policies | 4-13 |
| Common control properties | 4-14 |
| History policies | 4-16 |
| History import and export | 4-16 |
| <i>History import</i> | 4-17 |
| <i>History export</i> | 4-18 |
| Schedule import and export | 4-18 |
| Types of alarm service communication controls | 4-19 |
| Time sync service | 4-20 |
| Email service | 4-21 |
| Platform connection communications | 4-22 |

PREFACE

Preface contents

- [Document change log](#)
This section provides a list of significant changes to this document, listed in order of document publishing and revision date.
- [About this document](#)
This section describes this document in terms of its purpose, content, and target audience.
- [Related documentation](#)
This section contains a list of other relevant NiagaraAX documentation.

Document change log

Updates (changes and additions) to this document are listed as follows:

- *NiagaraAX Networking and IT Guide*: October 9, 2006
Initial Release

About this document

This document is intended to help you understand the parts that comprise the NiagaraAX framework product and help you understand how to integrate NiagaraAX devices into your network environment efficiently and securely. Included are topics that describe the NiagaraAX architecture in general, descriptions of many of the standard networking technologies and how they relate to NiagaraAX. This document also includes discussion and examples of networking security strategies that may be appropriate for your network application. This includes standard security practices and technologies, as well as descriptions of how NiagaraAX provides application security.

The following people should use this document:

- Systems integrators and installers
- Application programmers
- Corporate IT managers or IT personnel

Although this document includes basic descriptions of some networking principles and technologies, it is assumed that the reader has a basic understanding of NiagaraAX and networking in general.

It is important to use professional network security personnel to make sure that your network is secure when exposed on the Internet. It is also important to communicate and coordinate with the Information Technology (IT) department of the organization that owns the network resources you are using.

Related documentation

The following documents contain important information that pertains to NiagaraAX:

- *NiagaraAX User Guide*
- *NiagaraAX Developer Guide*
- *JACE NiagaraAX Install & Startup Guide*

CHAPTER 1

NiagaraAX introduction

The NiagaraAX framework includes integrated network management tools and a comprehensive toolset that enables non-programmers to build rich applications in a drag-and-drop environment. The primary characteristics of the NiagaraAX Framework are listed and described below:

Java-based The NiagaraAX framework uses the Java VM as a common runtime environment across various operating systems and hardware platforms. The core framework scales from small embedded controllers to high end servers. The framework runtime is targeted for J2ME compliant VMs. The user interface toolkit and graphical programming tools are targeted for J2SE 1.4 VMs.

Heterogeneous system integration NiagaraAX is designed from the ground up to assume that there will never be any one “standard” network protocol, distributed architecture, or fieldbus. NiagaraAX's design center is to integrate cleanly with all networks and protocols. The Niagara Framework standardizes what's inside the box, not what the box talks to.

Programming for non-programmers Most features in the Niagara Framework are designed for dual use. These features are designed around a set of Java APIs to be accessed by developers writing Java code. At the same, most features are also designed to be used through high level graphical programming and configuration tools. This vastly increases the scope of users capable of building applications on the NiagaraAX platform.

Embedded systems NiagaraAX is designed to run across the entire range of processors from small embedded devices to server class machines. Niagara is targeted for embedded systems capable of running a Java VM. This excludes some very low-end devices that lack 32-bit processors or have only several megabytes of RAM, but opens up a wide range of processor platforms. Embedded systems with the power of low end workstations have special needs. They have no integral display monitor and require remote administration. Embedded systems also tend to use solid state storage with limited write cycles and much smaller volume capacities than hard drives.

Distributed systems The framework is designed to provide scalability to highly distributed systems composed of 10,000s of nodes running the Niagara Framework software. Systems of this size span a wide range of network topologies and usually communicate over unreliable Internet connections. NiagaraAX is designed to provide an infrastructure for managing systems of this scale.

NiagaraAX architecture

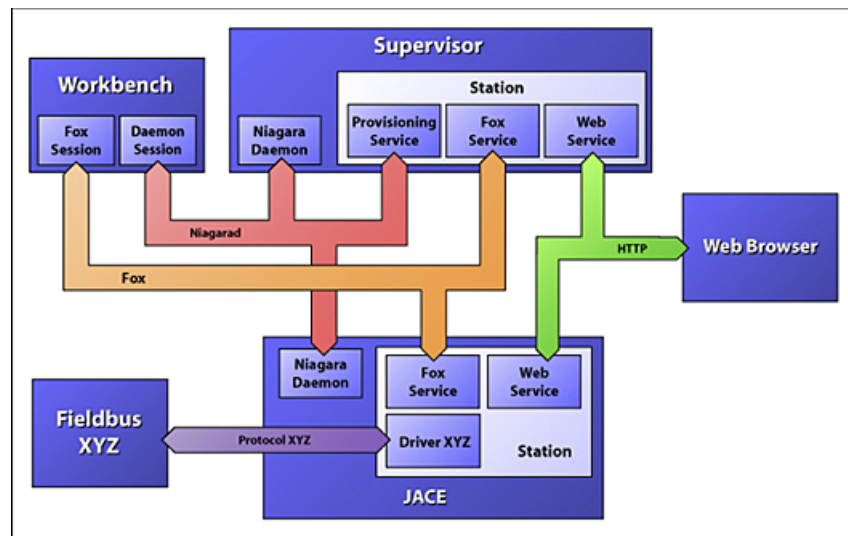
The following topics provide an introduction to NiagaraAX, including descriptions of some of the basic terms and concepts that are associated with the NiagaraAX architecture.

- [Types of NiagaraAX programs](#) (or processes)
- [Types of NiagaraAX protocols](#)
- [Types of NiagaraAX platforms](#)

Types of NiagaraAX programs

There are typically four different programs (or processes) associated with a NiagaraAX system. These programs and their network communication are illustrated in the [Figure 1](#)

Figure 1 NiagaraAX communications



Station: This is the NiagaraAX runtime – a Java VM which runs a NiagaraAX component application. Often the term Supervisor or JACE will be used interchangeably with “station.” Technically, the term station describes the component runtime environment common all to all platforms, and Supervisor and JACE describe the hosting platform.

Workbench: This the NiagaraAX engineering tool – a Java VM which hosts NiagaraAX plugin components.

Daemon: This is a native daemon process. The daemon is used to boot stations and to manage platform configuration such as IP settings. On Windows platforms, the daemon is run in the background as a Window’s service. On QNX it is run as a daemon process on startup. The most common way to access daemon functionality is through Workbench. A connection to the daemon is established via the “Open Platform” command which opens a PlatformSession to the remote machine. Workbench provides a means for accomplishing platform tasks, such as the following:

- installing and backing up station databases
- launching and monitoring stations
- configuration of TCP/IP settings (QNX, Windows XP, Server 2003)
- installation and upgrades of the operating system (QNX only)
- installation and upgrades of the Java virtual machine (JVM)
- installation and upgrades of the NiagaraAX software
- installation of lexicons for localization
- installation of licenses

Web Browser: This is a standard web browser such as IE or FireFox that hosts one of NiagaraAX's web user interfaces. NiagaraAX provides both server side and client side technologies to build web UIs:

- On the server side, the WebService component provides HTTP and HTTPS support in a station runtime. The NiagaraAX WebService provides a standard servlet engine.
- There are two client side technologies provided by NiagaraAX to support the browser user interface (BUI). The first is Web Workbench, which allows the standard Workbench software to be run inside a web browser using the Java Plugin. Web Workbench uses a small applet (wbapplet) to download modules as needed to the client machine and to host Workbench shell. These modules are cached locally on the browser's hard drive.
- In addition to Web Workbench, a suite of technology called Hx is available. The Hx framework is a set of server side servlets and a client side JavaScript library. Hx allows a real-time user interface to be built without use of the Java Plugin. It requires only web standards: HTML (Hypertext Markup Language), CSS (Cascading Stylesheets), and JavaScript.

Types of NiagaraAX protocols

There are three primary types of network protocols that NiagaraAX uses to integrate the four programs described above and illustrated in [Figure 1](#):

Fox: is the proprietary TCP/IP protocol used for station-to-station and Workbench-to-station communication.

HTTP: is the standard protocol used by web browsers to access web pages from a station.

Niagarad: is the proprietary protocol used for Workbench-to-daemon communication.

In addition to these protocols, other protocols are available as drivers (for example, SNMP).

About the Fox protocol

The Niagara Framework includes a proprietary protocol called Fox which is used for all network communication between stations as well as between Workbench and stations. Fox is a multiplexed peer to peer protocol that sits on top of a TCP connection. The default port for Fox connections is 1911.

Fox features include:

- Layered over a single TCP socket connection
- Digest authentication (username/passwords are encrypted)
- Peer to peer
- Request / response
- Asynchronous eventing
- Streaming
- Ability to support multiple applications over a single socket via channel multiplexing
- Text based framing and messaging for easy debugging
- Unified message payload syntax
- High performance
- Java implementation of the protocol stack

Types of NiagaraAX platforms

NiagaraAX is hosted on a wide range of platforms from small embedded controllers to high end servers. Instances of NiagaraAX software that run on these platforms are referred to as “stations.” Depending on the platform used, a station can offer different features and capabilities and be complemented with different options. In describing the NiagaraAX platforms, the following information also provides a general overview of the NiagaraAX product model (see [Figure 2](#)) and the various standard functions and options available.

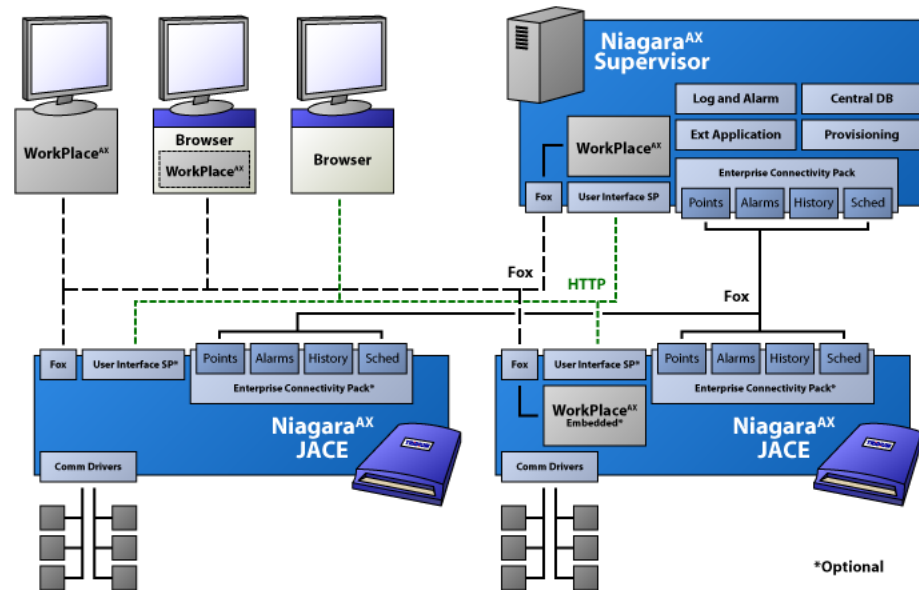
The NiagaraAX installation may consist of one or more of the following devices that may connect to or communicate on a NiagaraAX installation network:

- JACE Controllers
- NiagaraAX Supervisor
- WorkPlaceAX (and platform tools)
- Browser User Interface (BUI)

JACE: the term JACE (Java Application Control Engine) is used to describe a variety of dedicated host platforms. Typically a JACE runs on a Flash file system and provides battery backup. JACEs usually host a station and a daemon process, but not the NiagaraAX Workbench. JACEs typically run QNX or embedded Windows XP as their operating system.

Supervisor: the term Supervisor is applied to a station running on a workstation or server class machine. Supervisors are typically stations that provide support services to other stations within a system such as graphics, history, or alarm concentration. Supervisors by definition run a station, and may potentially run the daemon or Workbench.

Client: most often clients running a desktop OS such as Windows or Linux access NiagaraAX using Workbench or a web browser.

Figure 2 NiagaraAX product model

Types of JACE controllers

JACE (Java Application Control Engine) controllers are dedicated host platforms that provide integrated control, supervision, and network management services for networks of building monitoring and control devices. When JACEs are connected over an ethernet network using TCP/IP, they can communicate with each other on a peer-to-peer basis as well as communicate with other ethernet-based devices. Using the User Interface Station Pack (UI-SP), the JACE can also serve dynamic displays of the information contained in the connected devices to any standard web browser. JACE controllers run either the QNX operating system or a Win32 operating system. JACEs include the following:

- **JACE-2**
This device is a compact embedded controller and server platform with battery backup. It runs QNX RTOS, IBM J9 JVM, and NiagaraAX.
- **JACE-4 and JACE-5**
These devices are compact embedded processor platforms with flash memory, running QNX RTOS.
- **JACE NX**
This device is a compact PC with a conventional hard drive that runs an embedded version of Microsoft Windows XP and the Sun Hotspot VM.
- **SoftJACE**
The AX SoftJACE (Java Application Control Engine) is NiagaraAX Framework software that runs on a user-supplied PC, providing many of the same benefits as a JACE controller. The PC must meet the minimum requirements stated in the AX SoftJACE data sheet (Windows XP OS), and be dedicated to the SoftJACE application (single-purpose host).

NiagaraAX Supervisor

This “device” is a network PC acting as a server for multiple connected JACE stations. The AXSupervisor has the following features:

- Provisioning of multi-JACE systems (tools for updating and installation of software modules)
- Support for integration with standard RDBMS (MS SQL, Oracle, DB2, etc.)
- Platform for enterprise applications (Vykon Energy Suite) and others in the future
- Central database storage for attached JACEs
- Archive destination / repository for log and alarm data
- Central server of graphics and aggregated data (single IP address)

WorkPlaceAX (and the NiagaraAX platform)

This is the engineering tool that is used to create applications by defining components and linking them together to create logic and displays. It allows the user to develop comprehensive applications for control, monitoring, alarming, data logging, reporting, and real-time data visualization using a single graphical tool. WorkplaceAX can run as a standalone application on a PC, can be bundled with an AXSupervisor, or be served up to a browser from an embedded JACE platform.

“Platform” is the name for everything that is installed on a NiagaraAX host that is not part of a NiagaraAX station. The platform interface provides a way to address all the support tasks that allow you to setup and support and troubleshoot a NiagaraAX host. The platform daemon is a compact executable written in native code, meaning the daemon does not require the NiagaraAX core runtime, or even a Java VM. The platform daemon is pre-installed on every JACE controller (even as factory-shipped), and runs whenever the JACE is booted up.

Platform tools require a platform connection, which is different from a station connection. When connected to a NiagaraAX platform, Workbench communicates (as a client) to that host's platform daemon (also known as “niagarad” for Niagara daemon), a server process. Unlike a station connection, which uses the Fox protocol, a client platform connection requires Workbench, meaning it is unavailable using a standard Web browser.

A NiagaraAX host's platform daemon monitors a different TCP/IP port for client connections than does any running station (if any). By default, this is port 3011. Finally, the platform daemon uses “host-level” authentication for signon access. This means a user account and password separate from any station user account, and should be considered the highest level access to that host.

Browser user interface

The term “browser user interface” or “BUI” simply indicates user access of a NiagaraAX station (JACE controller or AXSupervisor) using a standard web browser. The BUI interface provides remote administration and monitoring of building control systems on an intranet or over the Internet.

CHAPTER 2

NiagaraAX and networking

About IP addressing

In the most widely installed level of the Internet Protocol (IP) today, an IP address is a 32-bit number that identifies each sender or receiver of information that is sent in packets across the network. IP addresses are actually made up of two parts, the network portion (identifies a particular network) and the host portion (identifies a particular device).

The network portion is used to determine whether other IP addresses are on the local network or a remote network. On the network, between the router that moves packets from one point to another along the route - only the network part of the address is looked at. All hosts on a given network share the same network number and can communicate with one another without being routed through an IP router.

In addition to the network address or number, information is needed about which specific machine (or host) on the network is sending or receiving a message. So the IP address needs both the unique network number and a host number (the host number must be unique within the network). The host number is sometimes called a local or machine address.

All NiagaraAX devices are configured with an IP address. An IP address is assigned to a host by an administrator and is not hardware specific, but rather is configured through a software interface on each host.

Private IP addresses

There are three blocks of IP addresses set aside for use in private networks. This allows organizations to implement IP addressing without having to apply to an ISP for unique global IP addresses for every host. Because these addresses are blocked by the global Internet routing system (the routers will drop packets from these addresses), the address space can simultaneously be used by many different organizations. However, hosts using these private addresses cannot be reached directly from the Internet (nor can they communicate to the Internet). If your NiagaraAX devices are on a VPN or all on the same LAN/WAN, you can assign private IP addresses to them, as required. The following list includes the sanctioned private addresses for each class.

- Class A - 10.0.0.0 to 10.255.255.255
This provides one network of 16,777,216 hosts
- Class B - 172.16.0.0 to 172.31.255.255
This range provides 16 networks, each with 65,534 hosts
- Class C - 192.168.0.0 to 192.168.255.255
This address range provides 254 networks, each with 254 hosts

To use the security of private IP addressing and the flexibility of public addressing, NiagaraAX devices and networks may be configured to use network address translation with one or more of the following: proxy servers, firewalls, or routers.

Network address translation (NAT)

To overcome IP address limitations, the use of private addresses is commonly teamed with a technique called Network Address Translation (NAT) to provide access to the Internet by hosts that require it. Typically, some device (such as a router, firewall, or proxy server) has a supply of legitimate addresses and translates between a private address and a public one for a host that needs access to or from the Internet.

***Note:** Some administrators choose to implement IP addressing on their private networks using legitimate addresses (such as 205.254.1.0) that have not been assigned to them. They use NAT to translate between the legitimate external address and the illegitimate internal address. Depending on how the Internet connection works, this may cause problems in the event of a failure in the connection. It is always best to use private address ranges instead.*

See [Figure 3](#) for examples of private networks and routers using NAT to translate between private and public IP addresses.

IP routing and default gateway

Any NiagaraAX host on a network can communicate with another host on the same network. However, if a host wants to communicate outside its network, the administrator sets a default gateway on the host which is the IP address of a router used for communication with other networks.

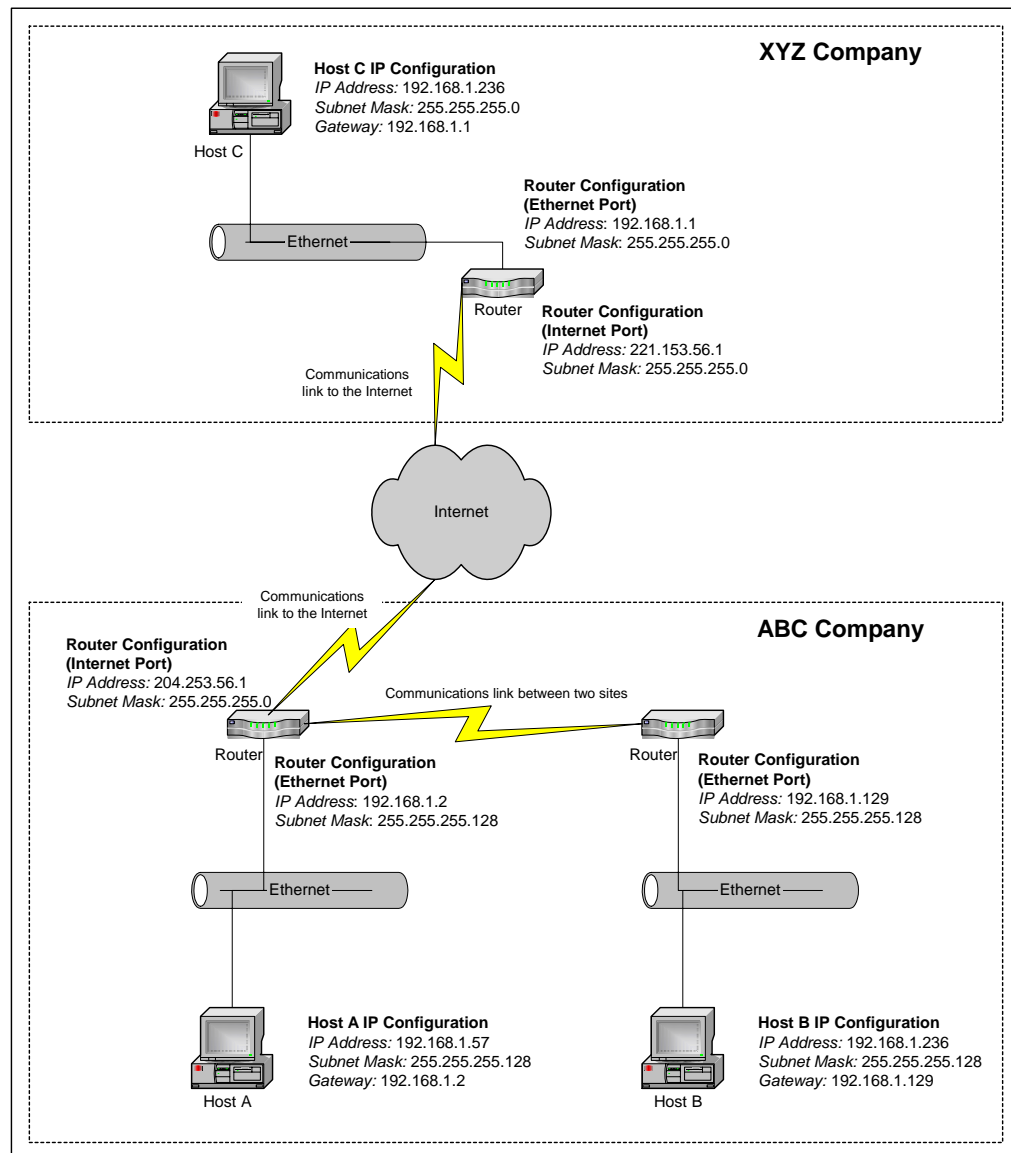
The router examines an IP packet from the host and compares the destination address with a table of addresses it holds in memory. If it finds a match, it forwards the packet to the address associated with the table entry. This address could be on another network attached directly to the router, or the router may forward it to another router that knows about the network of the destination address.

Routers build these tables of destinations in a number of different ways. For simple networks, the router may load a table during start-up that was manually created by the administrator. However, more typically, routers use a broadcasting protocol to advertise the networks that they know about. Routers use other protocols to discover the shortest path between networks (the least number of hops from router to router). Routes are updated periodically to reflect changes in the availability of a route.

Routing example

This example illustrates a typical network architecture of routers and default gateways in several networks. The following points are illustrated in [Figure 3](#)

- Company XYZ has a single network using the private class C address of 192.168.1.0.
- The router is performing NAT to translate the address of Host C to a legitimate address on the external network when Host C uses the Internet.
- The network is not subnetted.
- Company ABC uses the same private class C network, but has subnetted it to provide two networks, one at each site.
- The router is performing NAT to translate the addresses of hosts from both private networks to addresses on the external network when the hosts access the Internet.

Figure 3 Typical IP configuration

Static and dynamic IP addressing in NiagaraAX

Once the administrator has decided exactly what IP addressing scheme to implement, there are a number of methods to actually get the IP address (and subnet mask and default gateway information) onto the host. The simplest method is to configure this information manually on each machine, using the software provided by the operating system. However, this is quite time consuming in a network of more than a few hosts.

Most larger networks are made up of two types of devices: those that need a static (non-changing) IP address because they are accessed frequently by other devices, and most other hosts, which are rarely accessed by others. Hosts that are accessed

frequently (like servers and printers) are typically configured manually with a static IP address. The remaining hosts are configured to receive a dynamic address using a protocol such as Boot Protocol (BOOTP) or Dynamic Host Configuration Protocol (DHCP). Servers are set up with a pool of addresses to randomly distribute to clients (the host). They can also provide subnet mask, default gateway information, and DNS server information. The host is configured to request this information from the server using BOOTP or DHCP. When the host boots, it sends a broadcast message requesting an IP address, and a server responds with one of the numbers from the pool (and the remaining information). However, since the host is not necessarily guaranteed to receive the same IP address every time it boots, if a static IP is required, DHCP servers can be configured to provide a static IP address to a particular host.

You should provide a stable IP address for your NiagaraAX hosts. You can do this by making the address “static” (permanent) or by reserving the address on a DHCP server using the MAC address of the NiagaraAX host.

NiagaraAX and DHCP

DHCP is supported by the operating systems used by all NiagaraAX hosts, though static IP addresses provide the most reliable connectivity. To reliably use DHCP, it is better to reserve an IP address in the DHCP server for the MAC address of each NiagaraAX device. This ensures that the NiagaraAX device receives the same IP address whenever it requests one from the DHCP server.

Consider the following points concerning DHCP and NiagaraAX networking:

- Static IP addresses provide the most reliable connectivity between NiagaraAX hosts. Problems can develop, for example, if one station is assigned a new IP by the DHCP server and it has links to other stations. Those links stop working until those stations are restarted. The DHCP server should be configured to allocate the same IP address to the NiagaraAX host whenever it requests one. Note that this address is not really a static IP address, but rather is a dynamic one reserved by the DHCP server for the particular host.
- For the DHCP administrator to set up the reserved IP address, you will need to provide the MAC address of the NiagaraAX host to the DHCP administrator.
- You can set up DHCP on a JACE-4/5 or a JACE-2 using the TCP/IP Configuration view of the Workbench platform but you must use Windows networking tools to set up DHCP on any NiagaraAX host running the Windows OS.
- Windows 2000 and Windows XP DHCP servers can be configured to update DNS servers on behalf of devices that do not support dynamic update. NiagaraAX does not support Dynamic DNS.

About the HOSTS file

HOSTS files were the original mechanism used to resolve an IP address to a name. The HOSTS file is a text file residing on each local machine. Each line of the text file typically contains an IP address of a host and a name for it.

The primary limitation of this name resolution technique is that the HOSTS file is only usable by the machine on which it resides. For example, you could add an entry to Host A's HOSTS file mapping Host B's IP address to a name of "Pluto". Any application running on Host A could then contact Host B using the name "Pluto". However, Host B would not be addressable with "Pluto" by other hosts since the mapping only resides in Host A's local HOSTS file. That means another host wanting to contact "Pluto" would need a similar entry in its local HOSTS file. Therefore, HOSTS files are difficult to maintain when you have more than a few hosts, and other technologies such as DNS, DDNS and WINS are often used instead.

One of the advantages of using a HOSTS file is that it is not dependent on a server for name resolution, as is required in the other resolution protocols. For a higher degree of stability it is good to use a HOSTS file on all NiagaraAX hosts.

The HOSTS file is always the first place a host looks for name resolution, and if it finds an entry it uses it and does not check other name sources.

About DNS

The domain name system (DNS) is the mechanism used by hosts to resolve names on the Internet, and on some private networks as well. Hosts that participate in the DNS system have names like "www.tridium.com". This is referred to as the fully qualified domain name. These names must be globally unique so the data intended for that domain is accurately routed. To accomplish this, the domain names are controlled by ICANN, and governed like the IP address system with authority granted to accredited name registrars located throughout the world.

Domain names

As with IP addresses, the name is comprised of two parts: the domain name (tridium.com) and the host portion (for example, "www").

Name Servers

The first level of the domain naming system is organized into classes such as .info, .com, and .org, and countries, such as .uk, and .bm. Top-level domains are then divided into second-level domains such as va.us, co.uk, and tridium.com. A second level domain can then be subdivided into a third level (such as bbc.co.uk) and so forth. Domains can actually be subdivided into 127 levels, but it is rare to see a name with more than 4 levels.

NiagaraAX and DNS

NiagaraAX supports DNS as an alternative to the HOSTS file, although the most reliable connectivity to other NiagaraAX hosts is provided through the use of a HOSTS file on each NiagaraAX station. With HOSTS files, there is no dependency on a remote DNS server for name resolution.

Many host operating systems (OSs), support DNS. Administrators configure the host to look up entries in one or more DNS servers. The host can be configured manually, or it can dynamically receive this setting with DHCP.

When a host tries to contact a particular name (for example, trying to browse www.bbc.co.uk), it first looks in the local HOSTS file, and upon finding no entry, checks with its name server. The name server either returns the information if it knows it, or contacts a root name server, which passes it the address of one of the name servers responsible for the .uk domain. Then that server returns the address of a second-level server handling co.uk, which returns the address of the bbc.co.uk server, which finally returns the IP address for the host named “www”. The browser uses the IP address to fetch the data to open the page.

About WINS

WINS is Microsoft’s Windows-only name resolution protocol used by Windows machines prior to Windows 2000. Like DNS, WINS is also implemented with a series of servers that maintain databases of host names to IP addresses. However, setting up WINS is easier than setting up a DNS because WINS actually receives most address records automatically. Administrators name hosts and set one or more WINS server as part of the machine configuration. When a host boots it tells the WINS server its name and current IP address. However, the name that is registered is not in the DNS format of host.domain.xxx, but is in a proprietary format.

WINS is typically used within a company to provide host name resolution of company-specific Windows hosts. It is often teamed with DNS to provide external resolution.

A pre-Windows 2000 Microsoft host that needs to do a name lookup first looks it up on a WINS server, and if a match is not returned, then the DNS server is checked.

About DDNS

The dynamic domain name service (DDNS) allows a device with a dynamic IP address to be accessed using a well-known domain name. DNS servers handle mapping a name (such as MyJACE.EasyIP.net) to an IP address. With DDNS, each time a host receives a new IP address, it sends an update to the name-to-IP-address mapping on the DDNS server. The DDNS function can be implemented on hosts connecting to an internal LAN (with company DDNS servers) and the Internet (with third-party DDNS services). When configuring a JACE with captive ISP, many times you cannot obtain a static IP address for the JACE from the ISP. The DDNS may be implemented on Win32-based controllers with an Internet DDNS provider.

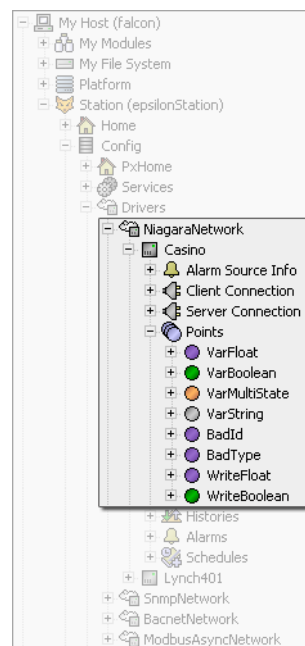
About the NiagaraAX network

Currently, by default, every NiagaraAX station has a *Niagara Network* under its *Drivers* container. The Niagara Network is where data is modeled that originates from other stations. Generally, this is done using the same driver architecture used by other (non-Niagara) drivers. However, a Niagara Network has the following unique differences:

- Data points under a Niagara Station are “read only” points of the types: Boolean Point, Enum Point, Numeric Point, or String Point.
- Connections between stations occur as client and server sessions using the Fox protocol. The requesting station is the client, and target (responding) station is the server. A Workbench connection to a station operates identically, where Workbench is the client, and station is server. Client authentication (performed by the server) is required in all Fox connections.

[Figure 4](#) shows how the NiagaraAX network is represented in Workbench.

Figure 4 Niagara Network in Workbench



About non-NiagaraAX networks

In any NiagaraAX station, one or more driver networks are used to fetch and model data values using proxy points, lower-tier components in that particular driver's architecture. To support a driver's proxy points, the station must have that driver's network architecture.

Networks are the top-level component for any NiagaraAX driver. For drivers that use field bus communications, such as Lonworks, BACnet, and Modbus (among many others), this often corresponds directly to a physical network of devices.

Often (except for BACnet), the network component matches one-to-one with a specific comm port on the NiagaraAX host platform, such as a serial (RS-232 or RS-485) port, Lonworks FTT-10 port, or Ethernet port.

***Note:** A BACnetNetwork component is unique because it supports multiple logical BACnet networks, which sometimes use different comm ports (e.g. if a JACE-4/5 with BACnet MS/TP, one or more RS-485 ports). See the BACnet Guide for more details.*

Other “non field bus” drivers also use a network architecture, for example the Ndio (Niagara Direct Input / Output) driver has an NdioNetwork to interface to physical I/O points on the host JACE (Ndio applies only to a JACE model with onboard I/O or with an attached hardware I/O board). Also, “database” drivers also use a network architecture, for example the “rdbSqlServer” driver includes an RdbmsNetwork. (Database drivers apply only to Supervisor or SoftJACE hosts.).

Within a driver architecture are a hierarchy of components that have properties that affect network activity. These component properties are used to monitor and control the network status (enabled, disabled, health, and similar items) and communication schedules, including polling updates and refresh rates.

Protocols

There are typically three network protocols that are used to integrate the four programs that are internal to Niagara (see [Types of NiagaraAX programs](#)):

Fox is the proprietary TCP/IP protocol used for station-to-station and workbench-to-station communication.

HTTP is the standard protocol used by web browsers to access web pages from a station.

Niagarad is the proprietary protocol used for workbench-to-daemon communication.

Niagara's design goal is to integrate cleanly with all networks and protocols. The Niagara software suite implements a highly efficient adaptation of the JavaBean component software model and Internet technologies to provide true interoperability across a wide range of automation products. The Niagara object model can be used to integrate a wide range of physical devices, controllers, and primitive control applications including LonMark profiles, BACnet objects, and legacy control points. The architecture supports future enhancements by allowing legacy systems to be brought forward, where they can readily adopt new standards, solutions, and applications.

Enterprise-level software standards include Transmission Control Protocol/Internet Protocol (TCP/IP), eXtensible Markup Language (XML), Hyper Text Transfer Protocol (HTTP), and others. These standards provide the foundation on which to build solutions that allow information to be shared between the control system and the enterprise information system.

Types of NiagaraAX hosts

NiagaraAX hosts may be classified as either QNX-based hosts or Win32-based hosts, depending on the operating system that they use. The following topics describe the characteristics of both host types:

- [QNX-based NiagaraAX hosts](#)
- [Win32-based NiagaraAX hosts](#)

QNX-based NiagaraAX hosts

Sometimes called “embedded” JACE controllers, these include JACE-2, -4, and -5 series controllers shipped with the QNX operating system. These devices use onboard flash memory for file storage. An option for an onboard dialup modem is available, or if needed, an external modem can be used.

The QNX-based controllers are not Windows hosts and therefore do not support WINS for name resolution of Windows hosts. They have the following characteristics:

- used to integrate legacy, LonWorks, and BACnet devices
- optionally provide browser user interface to NiagaraAX installation (JACE-2, JACE-4, and JACE-5 with UI option)
- support host name lookup on a DNS server
- can be a DHCP client
- cannot be a Windows DDNS or Active Directory client
- primary communication port is a 10/100mbps ethernet port with an RJ-45 connector. Other connections may include serial ports (RS-232, RS-485) and specialized ports, such as LonWorks FTT-10, depending on host.

All QNX-based JACE models have an integral backup battery. The backup battery allows continuous operation during brief power outages. The “JACE power monitoring” feature allows monitoring AC power and backup battery level, and a configurable delay for orderly shutdown of the JACE upon AC power failures. You can access power monitoring of a QNX-based JACE in the PowerMonitorService in a running station.

Win32-based NiagaraAX hosts

Win32 (Windows 32 bit) platforms include the JACE-NX, and any “Supervisor” PC host. As their operating system, Windows XP (if a Supervisor, possibly Windows 2000) is used, and file storage uses a hard drive. A JACE-NX has an option for an internal dialup modem, or it can use an external modem. If needed, a Supervisor PC's dialup modem can be an either internal or external type. They have the following characteristics:

- operate in a Windows environment just like any other Windows host
- used as a server for NiagaraAX integrations
- provides GUI to NiagaraAX integrations

- can be used to engineer and maintain NiagaraAX integrations
- can be a Windows DDNS client
- supports host name lookup on a DNS server
- can be a DHCP client
- primary communication port is ethernet; one or more RS-232 serial port connections

NiagaraAX host networking technologies

Networking technologies that are associated with NiagaraAX hosts, include the following:

Ethernet: QNX-based and Win32-based hosts Ethernet is required for NiagaraAX hosts for configuration.

TCP/IP (IPv4): QNX-based and Win32-based hosts TCP/IP is required for NiagaraAX hosts for configuration. TCP/IP use includes:

- IP Address (x.x.x.x)
- Subnet Mask
- Default Gateway

Public IP Address: QNX-based and Win32-based hosts Public IP Address is required for any NiagaraAX host that must be reached from the Internet.

DHCP: QNX-based and Win32-based hosts DHCP is supported on these hosts, however, in most NiagaraAX architectures, hosts require a static IP address to facilitate data passing.

HOSTS file: QNX-based and Win32-based hosts HOSTS files are local to each host, so they do not require dependency on a remote server for name resolution. Therefore, they are the recommended method for name resolution for NiagaraAX hosts.

DNS: Win32-based hosts DNS is supported in NiagaraAX, Win32-based hosts. However, the most reliable connectivity to other NiagaraAX hosts is provided through the use of a HOSTS file on each NiagaraAX station.

WINS: Win32-based hosts WINS is supported in NiagaraAX, Win32-based hosts. However, the most reliable connectivity to other NiagaraAX hosts is provided through the use of a HOSTS file on each NiagaraAX station.

DDNS: QNX-based and Win32-based hosts DDNS is available for NiagaraAX, Win32-based hosts (using Windows DDNS) and for QNX JACEs using a third party DDNS such as tzo.com.

Network Address Translation (NAT): QNX-based and Win32-based hosts Network Address Translation is supported on all NiagaraAX host operating systems.

Proxy Servers: QNX-based and Win32-based hosts Communication via proxy servers are supported on all NiagaraAX host operating systems.

Firewalls: QNX-based and Win32-based hosts NiagaraAX hosts operate well in many firewall environments, provided that are configured properly.

NiagaraAX network examples

There are many different ways to design and configure NiagaraAX networks. It is possible for each individual networking environment to combine with the project requirements to present a unique challenge for maximizing network efficiency. In addition to understanding general networking principles, it is important to understand NiagaraAX networking.

Please note the following general information points about connecting NiagaraAX devices to a LAN or WAN:

- Connection between NiagaraAX hosts on a LAN/WAN will be faster and more reliable than connection via modem (either direct dial or through an ISP).
- Niagara proxy points are designed to be used across connections that are always available. The design assumes that connections between linked hosts go up or down infrequently. Therefore, proxy points should only be used on a LAN or WAN, and only on a LAN/WAN that provides reliable connection between hosts.
- Access to hosts with private IP addresses can be made from hosts on the same LAN/WAN. Any host (NiagaraAX or other) that needs to be accessed directly from the Internet must have a public IP address. External hosts can only access privately-addressed internal hosts through a virtual private network (VPN).
- When connecting to NiagaraAX hosts behind a firewall, certain ports need to be opened on the firewall for access by a BUI client or by an engineering station running Workbench or the platform tools. For more information on ports in the NiagaraAX environment, see [About NiagaraAX ports](#).
- The key to success in many installations is early involvement by the IT department at the site. If you are going to connect devices to their LAN/WAN you need to check with them before implementing to learn about their policies so you can follow them.

The following examples represent some of the major characteristics of typical NiagaraAX system architectures and provide some common practices for engineering NiagaraAX environments in both single-site and multiple-site applications.

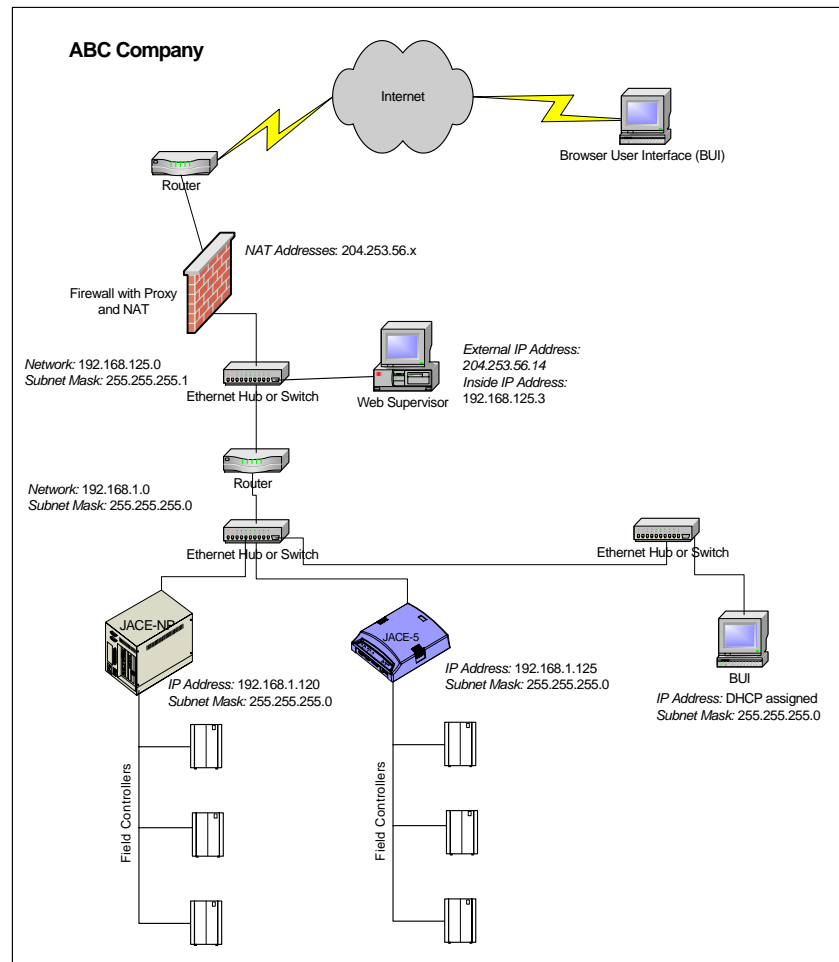
NiagaraAX single-site network application

In this example, a customer has a single site with a LAN, connecting to the Internet through a firewall (refer to [Figure 5](#)). The firewall provides security, as well as network address translation (NAT), which provides company devices with public IP addresses so that they are accessible from the Internet.

The site has multiple JACE controllers controlling field devices. These JACEs have private IP addresses so they are not accessible by the browser user interface (BUI) user located across the Internet. However, they are available to the BUI user located on the same LAN.

The AXSupervisor has a public IP address assigned to it in the firewall. It can be reached by the BUI user located across the Internet (the external user) and also the internal BUI user. In this example, the AXSupervisor has been engineered to include Px pages that show real-time information originating from the JACEs. To accomplish this, The AXSupervisor proxies data in the different JACE stations. In addition, the AXSupervisor functions as a supervisory station, archiving the other stations' data logs, alarms, and so on. This data is available to either BUI user.

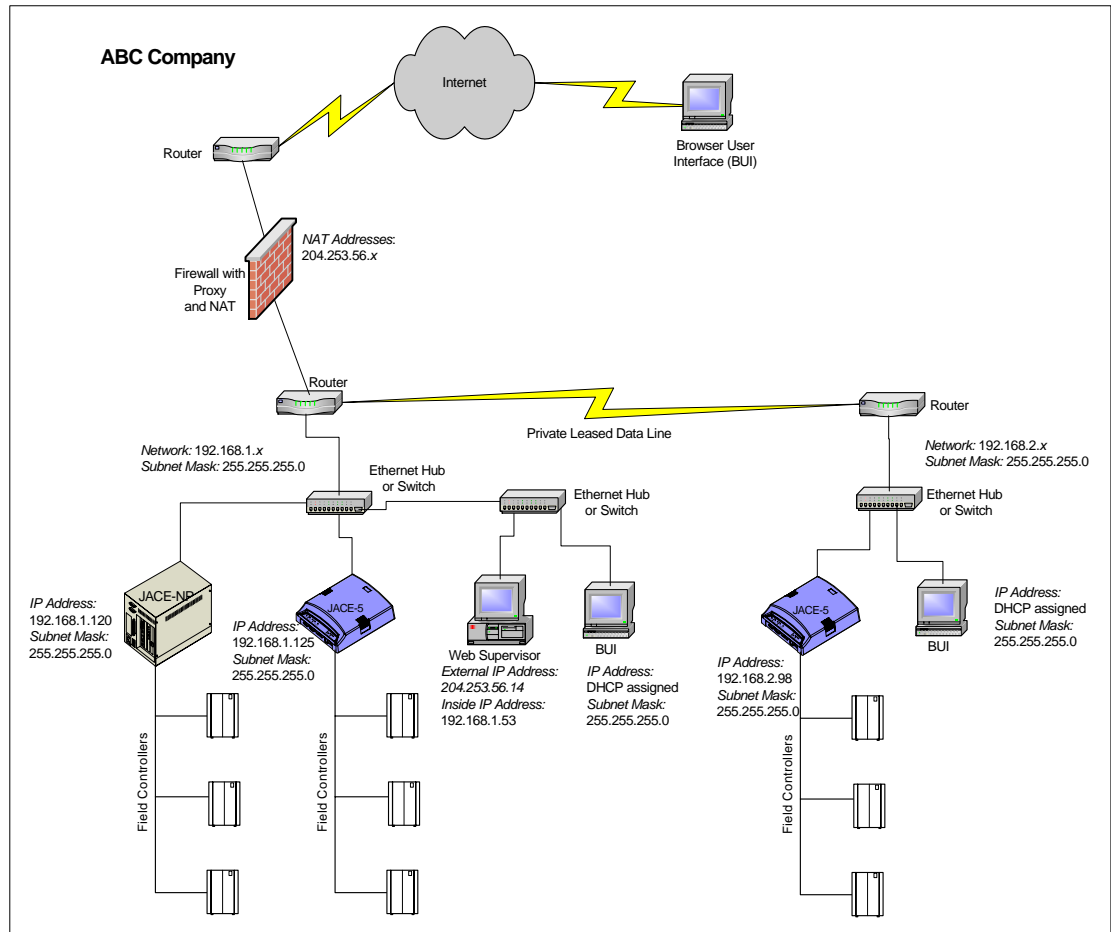
The network site administrator chooses to place the AXSupervisor outside the enterprise LAN but just behind the firewall. This allows faster access by the external BUI user because the network traffic between the external host and the AXSupervisor does not come onto the customer's enterprise LAN (which may be congested).

Figure 5 Typical single-site (LAN) architecture

NiagaraAX multi-site network application

In the example presented in [Figure 6](#), ABC Company has added a JACE to a LAN at another site. The sites connect to each other using a private data line that is leased from a phone company, thereby creating an enterprise-wide WAN.

Niagara proxy points are used to update the “live” graphic (Px) page presentations on the AXSupervisor. However, the network site administrator decided to move the AXSupervisor onto the enterprise LAN at the primary site, since there is more data traffic generated internally than the occasional external BUI use. The internal IP address of the AXSupervisor changed to one in the IP range of the new network to which it is attached, but the external address (because of NAT) did not change.

Figure 6 Typical multi-site (WAN) architecture

Managing NiagaraAX hosts

All of Tridium's NiagaraAX products can co-exist on a Windows 2000 or Windows XP infrastructure. Since all of the Win32-based controllers are built on the Windows platform, they appear in the Windows Network Neighborhood and can be browsed. At your discretion, these Win32-based devices can be members of the Windows domain or Active Directory.

The tools and methods that you choose to use for managing a NiagaraAX environment depend not only on the network design, but also on your access to the network and the tools that you have available to take advantage of that access. Moreover, each management task may be more specifically suited to one network management tool than another. Some of the specific NiagaraAX network management tools include the following:

- [AXSupervisor](#)
- [NiagaraAX platform](#)
- [Other tools](#)

AXSupervisor

You can design a NiagaraAX system so that it allows you to manage a facility's control equipment using a single AXSupervisor with a public IP address. Proxies of data in JACEs with private IP addresses to the exposed AXSupervisor can provide integration and aggregation of the information contained in the JACEs. However, if you need to maintain individual JACEs that are using private IP addresses on the network, you need to have access to at least one of the following:

- on-site network connectivity
- telephone line for direct dial into Windows RAS on a AXSupervisor or JACE. Once dialed in, you can reach the additional NiagaraAX devices.
- another non-NiagaraAX dialup solution into the network
- access to the AXSupervisor via remote control software. You can use Workbench on the AXSupervisor to manage the other hosts.
- VPN connectivity

NiagaraAX platform

Platform is the name for everything that is installed on a NiagaraAX host that is not part of a NiagaraAX station. The platform interface provides a way to address all the support tasks that allow you to setup, support and troubleshoot a NiagaraAX host. The following list includes a summary description of what you can do with the platform, including the functions, views, and typical usage of the platform.

Dialup Configuration This view is used to configure the platform's dialup modem, including settings for a "captive network." If a QNX-based JACE, separate "listening" settings apply for receiving dialup connections.

User Manager This view applies to remote Win32 platforms (JACE-NX). To access host Windows OS user and group accounts, including ability to add or delete users/groups, change passwords and group members.

Distribution File Installer This view is used to compare the remote platform's NiagaraAX runtime environment (nre) to locally available distribution (.dist) files, and if desired, to update the remote platform. The nre is divided into two separate .dist files: a "config" and a "core," where the config is customizable after installation.

File Transfer Client This view is used to copy files between your Workbench PC and the remote NiagaraAX platform (in either direction).

Lexicon Installer This view is used to install NiagaraAX lexicon files from your Workbench PC to the remote NiagaraAX platform, in order to provide non-English language support.

License Manager This view is used to review, install, save, or delete licenses and certificates on the remote NiagaraAX platform.

Software Manager This view is used to review, install, update, or uninstall “installable software” on the remote NiagaraAX platform. This includes NiagaraAX modules (.jars) as well as separate .dist files for NRE core, Java VM, and (QNX) OS. The Software Manager compares software installed on the connected platform against that available (locally) on your Workbench PC.

Platform Administration This view is used to perform configuration, status, and troubleshooting of the NiagaraAX platform daemon. Included are commands to change time/date, backup all remote configuration, and reboot the host platform.

Station Copier This view is used to install (copy) a station from your Workbench PC and the remote NiagaraAX platform, including different file-level options. Also to backup (copy) a station in a remote platform to your Workbench PC, or to delete a remote station. You can easily rename any copied stations.

Station Director This view is used to start, stop, restart, or kill a station on the NiagaraAX platform. Output from the station displays in the view pane, useful for monitoring and troubleshooting. You also configure a station's “Auto-Start” and “Restart on Failure” settings from this view.

TCP/IP Configuration This view is used to review and configure the TCP/IP settings for the network adapter(s) of the NiagaraAX platform.

Remote File System This view is used to navigate among all files and folders under the platform's Niagara root (system home) directory, including the ability to make local copies on your PC.

About the platform connection

A platform connection is different than a station connection. When connected to a NiagaraAX platform, Workbench communicates (as a client) to that host's platform daemon (also known as “niagarad” for Niagara daemon), a server process. Unlike a station connection, which uses the Fox protocol, a client platform connection requires the full Workbench application, therefore it is unavailable when you are running Workbench in a standard web browser.

The platform daemon is a compact executable written in native code, meaning the daemon does not require the NiagaraAX core runtime, or even a Java VM. The platform daemon is pre-installed on every JACE controller (even as factory-shipped), and runs whenever the JACE has booted up.

Also, a NiagaraAX host's platform daemon monitors a different TCP/IP port for client connections than does any running station (if any). By default, this is port 3011. Finally, the platform daemon uses “host-level” authentication for signon access. This means a user account and password separate from any station user account, and should be considered the highest level access to that host.

Types of platform administration functions

The following list provides a summary description of platform administration functions that are available using the platform connection in Workbench:

- View Details
Provides platform summary data, available to the Windows clipboard.

Includes all summary information shown in main Platform Administration view, plus installed modules, and so on.

- **Update Authentication**
For dialog boxes to change platform login access (user name and password). In “QNX-based” platforms this is simple, as there is only one platform administrator. “Win32-based” platforms offer a choice of a single (digest) platform account, or use of Windows OS user accounts (basic authentication).
- **Change HTTP Port**
For a dialog box to change the HTTP port for the host's platform daemon from (default) port 3011 to some other port.
- **Change Date/Time**
For a dialog box to change the host's current date, time, and time zone, as used by that host's OS.
- **FTP/Telnet (QNX-based only)**
For a dialog box to enable/disable both FTP and Telnet access to the JACE, or change the default port number used by each one.
- **Change Output Settings**
Provides a dialog box to change the log level of different processes that can appear in the platform daemon output.
- **View Daemon Output**
Provides a window in which you can observe debug messages from the platform daemon in real time, including the ability to pause the output and stream output to a file.
- **Set Module Filter**
Provides a dialog box to change the module content level of the host. Used very infrequently.
- **Backup**
Make a complete backup of all configuration on the connected host platform, including all station files as well as other NiagaraAX configuration.
- **Commissioning**
One way to launch the Commissioning Wizard, as an alternative to right-clicking on Platform in the Nav tree.
- **Reboot**
Provides a method to reboot a JACE platform, which restarts all software including the OS and JVM, then the platform daemon, then (if so configured in the Station Director) the installed station. If you click this, a confirmation dialog box appears. If you answer yes, the JACE is rebooted and the platform connection drops.

About the platform daemon on a PC

When you install NiagaraAX on your PC, you have the option to install and start Platform Daemon. The default selection is to install.

Important: *It is important to understand that you need the platform daemon locally installed and running in order to do any of the following:*

- host a NiagaraAX station on your local PC, such as for a Supervisor. This lets you open a Workbench client platform connection to your local (“My Host”) platform.
- use Workbench to open a remote JACE platform (or station) using a dialup connection, that is, using your PC's modem. (The platform daemon is not used to open a LAN-connected platform).

Once installed and started on your PC, you can see the platform daemon listed as a Niagara service from your Windows Control Panel, by selecting Administrative Tools->Services.

***Note:** Following NiagaraAX installation on your PC, you can alternately install and start the platform daemon at a later time, if needed. Under the Windows Start menu, do this by selecting Start->All Programs->Niagara <3.x.x>->Install Platform Daemon*

In summary (except for dial-out support), your PC's local platform daemon is not used in making client connections to other NiagaraAX hosts, only to provide the ability to run a station locally.

Other host administration tools

Following, is a list of some of the other NiagaraAX tools that are available for host administration tasks.

Platform Services This service allows access from a running station—in other words, from the station's perspective on its host platform. Unlike the various other platform views, a platform connection is not needed to access the platform services. Instead, you need only a standard station (Fox) connection. Platform services do not provide the full set of functions available in Workbench, as when you have a platform connection. For example, you cannot install or upgrade software, or transfer stations and files. However, a number of important configuration views for the platform are made available under a station's PlatformServices.

Included under PlatformServices are a TcpIpService and LicenseService, providing station (Fox) access to dialogs used in platform views, for instance the TCP/IP Configuration dialog box. These services support installations where all configuration must be possible using only a browser connection (and not Workbench connected to the JACE's platform daemon). Also included under PlatformServices, for any QNX-based JACE, is a serial port and power monitoring service.

System shell Any QNX-based JACE (JACE-2, -4, and -5 series) has a “system shell,” providing low-level access to a few basic platform settings. Using a special JACE power-up mode, you can access this system shell using a serial connection to the JACE's onboard RS-232 port. Typical usage is for troubleshooting. However, in the case of IP address mis-configuration, you may need to use this shell in order to regain access to the unit.

Tip: Depending on your preference, as an alternative to reconfiguring your PC's IP address in Windows (to initially connect to a new JACE), you may wish to use the serial system shell to set the JACE's IP address. If done as the first step, afterwards you could connect normally (Ethernet/IP) and perform all other NiagaraAX software installation/platform configuration using Workbench and the Commissioning Wizard. This method would save you from having to re-configure your PC's IP address settings in Windows: first to connect to the JACE as shipped from the factory, and then back again to its original settings.

Serial tunneling A NiagaraAX station running one or more serial-based drivers can provide “tunneling” access to its connected devices. This allows you to use a vendor's Windows serial-based application (via the serial tunnel client) to perform device-specific operations. Examples include application downloads or other device configuration.

The tunneling client is separate from NiagaraAX Workbench—meaning that you can install it on various PCs, as needed. The key advantage is that serial tunneling requires only a standard IP connection (to the station), yet provides access as if the client PC was attached to the target serial network via a physical COM port, for example RS-232.

Note: No special licensing is required to use tunneling features in NiagaraAX.

NiagaraAX backups

A NiagaraAX station includes a Backup Service. The BackupService provides for complete configuration backup of a JACE to your local Workbench PC or a to a browser PC (for users with Wb web or default Hx profile). By default, the BackupService is included when you create a new station using the New Station wizard. The remote Niagara host (JACE, Supervisor) must have the backup module installed.

A backup is a dist file (a zipped format) including, minimally, the station's config.bog, current station console output (.txt file), and backup.log file. If other station file types and subfolders are not excluded (in the “BackupService configuration”), the backup dist file contains them too—for example, files of type: Px, nav, html, jpg, png, and so forth. Also, the backup dist contains the zipped “nre-config” for that host (including license and certificates files), as well as pointers to the installed “nre-core,” OS, and JVM, each by version. Essentially, a backup dist provides a “configuration snapshot” of the entire JACE platform in zipped “dist” file format. This allows for a complete image restoration.

Note: Not included in a backup is runtime data that is actively managed by the station, such as the alarm and history databases. This data should be “backed up” using standard alarm routing and history archiving to a Supervisor host.

To restore a backup dist from Workbench, you open a platform connection to the JACE, then use the platform Distribution File Installer to install. Restoring from a backup may be necessary if the host hardware was replaced.

NiagaraAX hosts and network communication

In a NiagaraAX installation there is a wide variety of possible communications between NiagaraAX hosts. The following list presents common types of NiagaraAX host network communication and a description of the communication:

Browser User Interface The BUI is used to initiate communication from any client host to any NiagaraAX host for the purpose of:

- viewing real-time control information
- maintaining a station (through the use of servlets)
- viewing station data (logs, alarms, schedules, and other information)

Station and host administration These are communications that are typically initiated by a Systems Integrator (SI) from an AXSupervisor or remote PC using Workbench or platform tools. The receiving host (or server) could be any NiagaraAX host. The communication could be initiated to perform any of the following tasks:

- create a new station
- maintain or administer a live station
- change host configuration
- add users to the host
- install a station
- install software or licenses
- perform database administration

Archiving this is communication that is initiated by any NiagaraAX host that is set up to archive logs (histories). These logs are either “pushed” (exported) to or “pulled” (imported) by a remote AXSupervisor for long term storage.

Alarming archiving This is communication initiated from any NiagaraAX host that is set up to remotely archive alarms. These alarms may be sent to any AXSupervisor that is set up to archive the alarms. Usually the recipient host runs the Alarm Console for the purpose of acknowledging the alarms.

Alarm console acknowledgement This is communication initiated from an AXSupervisor to acknowledge an alarm on the NiagaraAX host that is archiving the alarm.

Global data passing This is communication that may be initiated from any NiagaraAX host to any other NiagaraAX host for the purpose of exchanging real-time data via the Niagara proxy points.

Station monitor This communication is a timed ping that may be initiated from any NiagaraAX host to the IP address of any other NiagaraAX host for the purpose verifying network connectivity. If the ping fails over a specified time, the initiating host generates a station alarm.

Other Other communications could include the following:

- Telnet, FTP, or Hyperterminal communications from Win32-based hosts to QNX-based hosts, as “alternative” means to accomplish host or station maintenance.
- Email alarm notification (notifications or acknowledgements) from any NiagaraAX host that is running the email service to any SMTP mail server.
- Remote printer alarm notification from any AXSupervisor to a networked printer.
- Time synchronization from any NiagaraAX host with the TimeSync service to a time server in order to synchronize the host time. The server could be a NiagaraAX host providing this function or any other server running the Internet Time Protocol.
- Station backups using a connection from a NiagaraAX Workbench that has the backup service installed. The target NiagaraAX host (JACE, Supervisor) must have the backup module installed.

CHAPTER 3

NiagaraAX security

NiagaraAX security model

NiagaraAX includes a comprehensive security model that provides a high degree of flexibility in managing access privileges to a NiagaraAX station (a JACE, AXSupervisor, or a Workbench-based tool such as WorkPlaceAX). The security model addresses both the platform connection and a station connection.

Platform connection Platform connection refers to a connection from a Workbench-based engineering tool (e.g., the Vykon WorkPlaceAX) to the host's "platform daemon". The platform daemon is a compact executable that is pre-installed on every JACE controller. It provides support for tasks that allow setup and management of a NiagaraAX host. The platform daemon monitors a different TCP/IP port for client connections than does a NiagaraAX station and uses "host-level" authentication for access. Host-level access is controlled by a set of user accounts and passwords which are separate from any station user account and are part of the host operating system. On an AXSupervisor running on a Windows host, for example, platform user accounts would be part of the Windows user accounts.

The host-level user account provides Administrator Level access making the platform connection the highest level of access to a NiagaraAX host. It is used to copy a new station database to the host, or start and stop the station on the host, among other management functions. Note that the actual user name and password for the platform daemon account can be same as a station user name and password, but the accounts are different. For example, the user would need to enter the user name and password into the logon dialog for the platform even if that user was logged into the station.

Station connection Station connection refers to a connection by a person using either a Workbench-based tool or a web browser to a NiagaraAX station, or connection by one station to another station. The NiagaraAX security mechanism determines if a station connection is allowed, what can be accessed, and what operations can be performed.

The NiagaraAX security model is based on the concept of Users, Permissions, and Categories. Anything that needs to be protected (components, files, histories) is grouped into one or more categories. Each user is granted a set of permissions in each category. This combination of categories and permissions then defines exactly what each user can do with each object in the system. The following sections explain the various aspects of NiagaraAX station security:

Users Typically a “User” represents a human user who needs to connect to a NiagaraAX station, but it can also be used to represent machine accounts for machine to machine (station to station) connections. Every station has two built-in users that cannot be deleted or renamed: “admin” and “guest”. User “admin” is always a super user, meaning it has all permissions in every category and can thus access everything in a station. User “guest” is disabled by default. If enabled it provides station access from a web browser with no authentication (user is not prompted to login). User “guest” can then navigate to any object that has been assigned read permission for the user account “guest”. Every user of the system should be given a unique user name and password to make the audit logs more valuable.

Users have properties that define how the user navigates around the station, their language preferences, time and unit preference, etc. The main properties of a User related to security and station connection are their login credentials (e.g., user name, and password). Whenever a connection attempt is made, these credentials are checked against the users that have been configured in the station. This process is called Authentication (more detail is provided on this topic below).

Users are stored in the station's local database by default and looked up by the station's User Service. Alternatively, users can be looked up using the Lightweight Directory Access Protocol (LDAP). LDAP provides a central location for administering users of a NiagaraAX system. In addition to storing the login credentials, a user profile can be stored in the LDAP server. This user profile can then be matched to a “prototype user” in the NiagaraAX system to grant the user who is logging in the same privileges as the prototype user. For example, a NiagaraAX station could be configured with a prototype user called “Engineering”. This user “Engineering” could then be assigned as a profile to User1, User2, etc. in the LDAP server. When any user who has been assigned this “Engineering” profile in the LDAP server is logged into the NiagaraAX station and authenticated via LDAP, they will have the same access rights and privileges that the “Engineering” user has in the NiagaraAX station.

Authentication Whenever a station connection attempt is made, the user's login credentials are authenticated. There are three authentication points in the NiagaraAX framework:

- Workbench-to-station via the Fox protocol:
The user is prompted for a user name and password which is used to authenticate the Fox connection. The default authentication mechanism is Digest authentication which encrypts the password so that it is not passed in clear text. If LDAP is used then basic authentication must be configured between Workbench and the station because the station needs the password in text to pass on to the LDAP server. Secure Sockets Layer (SSL) can be used between the station and the LDAP server to connect to the SSL port of the LDAP server.
- Station-to-station via Fox:
The concept is the same as Workbench-to-station connection but instead of prompting for a user name and password, a pre-configured user name and password stored under the proxy for the station being connected to is used. This pre-configured user name and password (typically with super user

permissions) must correspond to a valid user in the station being connected to.

– Web browser-to-station (HTTP):

The user is prompted for a user name and password unless the “guest” user account is enabled. The default authentication mechanism is a cookie. Note that in a multi-station installation if the NiagaraAX stations are installed on a DNS domain and accessed using the full DNS name, you can configure stations so that a browser user is authenticated only once (upon initial station connection). In this scenario the same credentials used in the initial login are used in other station connections that the user navigates to via hyperlinks from the original station.

Secure Connections via SSL NiagaraAX supports SSL (Secure Sockets Layer) to provide secure communications between a web browser and the station. SSL works by using a private key to encrypt data that's transferred over the SSL connection between the web browser and the NiagaraAX station thus providing privacy, authentication and message integrity over the public Internet. URLs that begin with HTTPS indicate that an SSL connection will be used. HTTPS can be enabled from the Web Service in the NiagaraAX station if required.

Categories A Category is simply a name for a logical grouping of items or components. Categories are typically named to reflect what the grouping contains. For example, the “Lighting” category group might contain objects related to lighting and the “Floor 1” category might contain objects related to Floor 1. Any object that needs to be protected with individual security rules can be assigned to one or more categories. If an object is not explicitly assigned to any category it inherits the categories of its parent.

Objects store the categories they belong to as a variable length bit string so they can belong to as many categories as needed. Since each object stores the categories it belongs to the NiagaraAX security model provides excellent performance and scalability.

Permissions Permissions define what rights a user has within each of the categories in the station. There are two NiagaraAX permission levels: operator and admin. Within each level, separate options exist for read access, write access and invoke (action) access.

Every user in the station is configured with a permissions map. This map grants the user permissions for each category defined in the station, thereby granting the user permissions for objects assigned to that category. This provides tremendous flexibility. For example, if a user lacks read permissions on any object, that object is not be visible in the client display. If that object contains other components, then none of the child components are visible (regardless of their individual permission level). If a user has read permission on a component but only has read permission of *some* of its child components, then all of those child components are visible but access to the properties and any lower-level components are prohibited on those child components lacking read privilege. Super users are automatically assigned every permission in every category for every object.

NiagaraAX and Niagara R2 security comparison

Niagara R2 provides a robust security model. NiagaraAX builds on the R2 security model and provides better performance and scalability. Niagara R2 allowed for only eight “Security Groups”. NiagaraAX provides unlimited (limited by host memory of course) categories.

Also, NiagaraAX provides the following additional features related to security:

- Digest authentication
- LDAP support
- HTTPS support
- Single signon from a web browser if using DNS configuration
- User-friendly graphical tools to manage security in a NiagaraAX system

NiagaraAX security considerations

Any host connected to the Internet is vulnerable to attacks by someone else in the Internet community. This is especially true of any host that stays connected to the Internet virtually full time. NiagaraAX hosts that may be vulnerable include:

- Any NiagaraAX host connected to a company’s LAN and which has a public IP address.
- Any NiagaraAX host directly connected to an ISP and which has a public IP address.
- NiagaraAX devices function as proprietary web servers, not typical client machines. As part of normal station operations, they do not download any files. However, you may want to install virus protection for an AXSupervisor PC if it is used for other (non-NiagaraAX functions).
- The NiagaraAX framework does not use the Microsoft IIS server, instead it is a pure Java web server developed by Tridium. This eliminates many security holes associated with the Microsoft IIS server. The Fox protocol is a proprietary HTTP protocol which makes it highly unlikely that someone could hack the system without reverse engineering the product.

Typically, Win32-based hosts are more vulnerable than QNX-based hosts. This is a function of two factors:

- in Windows, there are many access points open by default that attackers can exploit. In contrast, the QNX OS has fewer access points enabled by default.
- the widespread availability of the Windows OS itself. Because the QNX OS is less common, people have not taken the time to figure out how to attack it.

Another common point of attack for Internet hosts is the web server that runs on many Internet hosts (including NiagaraAX hosts). However, the NiagaraAX web server implementation is proprietary and not subject to the well-advertised attacks on Microsoft Internet Information Server and the Apache HTTP Server.

The following security suggestions are provided to help you secure NiagaraAX hosts when connecting them to the Internet. You should evaluate the suggestions to see if they are applicable for each job that you architect.

Note: Many of these suggestions are also good guidelines for connecting hosts even in a LAN/WAN or direct-dial environment. Anyone with physical (or network) access to a host can be considered a security threat. You may want to consider implementing some of these, regardless of Internet connectivity.

General Security Guidelines

- Architect a LAN/WAN-only or LAN/WAN plus direct-dial solution
The most obvious way to protect hosts is to avoid connecting them to the Internet at all. However, that limits connectivity from other hosts already connected to the Internet (typically BUI users or other NiagaraAX hosts).
- Implement a firewall between your NiagaraAX host and the rest of the Internet community. Firewalls provide a barrier between the Internet community and protected hosts
- Implement a VPN between NiagaraAX hosts
- Implement strong passwords on each NiagaraAX host and station
Implementing strong passwords may prevent an attacker from guessing a NiagaraAX host or station password
- Change the default administrator password or establish a new administrator account on each host and delete or disable the default one that ships with the product. Each JACE ships with at least one default host administrator user name and password (typically tridium/niagara). If you do not change or disable this account, any person familiar with NiagaraAX software can gain administrative access to the host.
- If you change the password (or create a new account and disable the default), be sure to record your changes and store them in a place you (and your colleagues) can find them again. If you forget or lose the name or password you must ship the unit back for recovery.
- Change the default HTTP port (and other ports)—Changing server-side ports keeps out novice attackers, but may not stop more sophisticated ones.

Guidelines for QNX-based NiagaraAX hosts

- Do not enable FTP or telnet—FTP and telnet are standard Internet protocols with well-documented attack points. If you must enable FTP or telnet on a QNX-based host, consider changing the port to keep the novice attacker out. This may not stop a more sophisticated attacker who uses port scanning software to learn about all the open ports on a host.

Guidelines for Win32-based NiagaraAX hosts

- Implement and maintain virus protection on any AXSupervisor that will connect to non-NiagaraAX resources on the Internet—If your AXSupervisor connects to other Internet resources (i.e., web pages, e-mail) you should

implement and maintain virus protection. Viruses can make your AXSupervisor inoperable.

- Do not share folders on any Windows-based host—Windows shares provide file-level access to the Windows host. Since they advertise themselves, once an attacker has deciphered a host password they are very vulnerable. If you must use Windows shares, make sure to assign permissions only to accounts using strong passwords.
- Implement any security patches available from the OS vendor—Be sure to periodically review and update patches when new vulnerabilities are patched. For more information on securing Windows-based hosts, see the Microsoft Security Resource Center at <http://www.microsoft.com/security/default.asp>.

Creating a strong password

The best secure password is difficult to guess and difficult to crack, but easy for you to remember. To provide the highest level of security, the password should challenge password cracking software so that it takes more time to crack than most people would be willing to devote to it.

The following types of passwords are insecure because they are easy to guess by people you know or easy to crack by people you do not know:

- Any word, common or not, even one in a foreign language
- Any name (yours, your spouse or children, your boss, your pet)
- Any password made up of all numbers (bank card number, house numbers, telephone numbers, US Social Security number, or car license plate number)

A secure password should contain at least three of the following elements:

- Have at least eight characters (the more characters, the longer it takes to crack)
- Have both upper and lower case letters
- Contain both letters and numbers
- Contain special characters (interspersed between letters and numbers)

In addition, a secure host password should also be easy for you to remember so that you are not tempted to write it down and leave it in an insecure area (such as taping it to the unit). As mentioned in the previous section it is a good idea to note any password (or name) that you create or change and keep it in a secure area that is still accessible to the rest of your team members.

Note: *One common method for creating a strong password entails swapping out alphabetic characters with numeric or special-character equivalents. For example:*

- the word baseball becomes b@s3B@11
- the phrase playmahjong becomes p!@yM@4j0nG

For more information on securing hosts on the Internet, see <http://www.cert.org/>.

Firewalls and proxy servers

Note: You should consult the system owner's IT department on these issues during the planning phase of the installation to avoid unnecessary delays and rework resulting from a lack of adequate communication.

Both JACE and AXSupervisor can use NAT (name / address translation) through a firewall to expose them to the Internet. The firewall should be used to filter traffic at the port level to any exposed NiagaraAX device.

NiagaraAX hosts function well in many firewall environments, with the following conditions:

- In order to use the BUI in some profiles, Java applets must be permitted to download through the firewall. Any NiagaraAX host serving up non-Hx pages in the browser must be able to send the applets associated with these servlet pages to a BUI client.
- BUI to station communications uses HTTP protocol.
- On any firewall, application ports may need to be opened to allow communication between any two NiagaraAX hosts on opposite sides of the firewall.
- The following connections are created using the Fox protocol. Proxy servers and firewalls need to allow Fox traffic to pass between NiagaraAX hosts to enable these features:
 - Niagara proxy points (station to station data exchange)
 - alarm console to monitor alarms on a remote station
 - station monitoring function to monitor a remote host
 - pushed (exported) or pulled (imported) archiving
 - using Workbench to engineer a remote station

Note: If you are implementing a firewall in lieu of using one already in place, a good security practice is to grant the most limited permissions you can on the firewall, while still following the guidelines for communication listed above.

About virtual private networks

An alternate method of securely connecting Internet-attached NiagaraAX hosts is through the use of a virtual private network (VPN).

A VPN is an encrypted IP connection between hosts over a public infrastructure such as the Internet or the public telephone network. A VPN embeds a special protocol within the TCP/IP packets carried over the Internet. This concept of a second network protocol within a primary protocol is called tunneling. The following tunneling protocols are commonly found in VPN installations:

- PPTP (point-to-point tunneling protocol)
- IPSec (IP security protocol)
- L2TP (layer 2 tunneling protocol)

Along with encryption, many VPNs also include strong authentication of remote users or hosts and ways to hide information about the private LAN from hosts on the public network. A VPN can be between an individual computer and a LAN or can be LAN-to-LAN. Many companies use a VPN for connecting traveling or teleworking users, or for connecting small, remote sites to the corporate LAN.

Typically, a VPN architecture is comprised of:

- a client running software that is configured with parameters such as server IP address and tunneling protocol. The client could be an individual workstation (for computer-to-LAN VPNs), or another router or server (for LAN-to-LAN VPNs).
- a server device that handles the client connection, authentication, and decryption of the information from the client. A VPN server could be part of a firewall, or be a separate device.

Some advantages of using VPNs include:

- the client actually becomes part of the remote LAN (it receives an IP address on the remote LAN) and therefore has access to any resources on the LAN.
- cost can be lower than direct-dial (no extra telephone lines, RAS equipment to maintain, or long distance charges).
- if using cable or DSL connection, transmission speed can be faster than using direct-dial.

Some disadvantages include:

- overhead makes any VPN connection slower than a native (no VPN) dialup or cable/DSL connection.
- host will not connect if the Internet connection to the ISP or from the ISP to primary site is down.

Example NiagaraAX VPN network

[Figure 7](#) provides examples of typical NiagaraAX job configurations (system architectures) for connecting NiagaraAX hosts with a VPN.

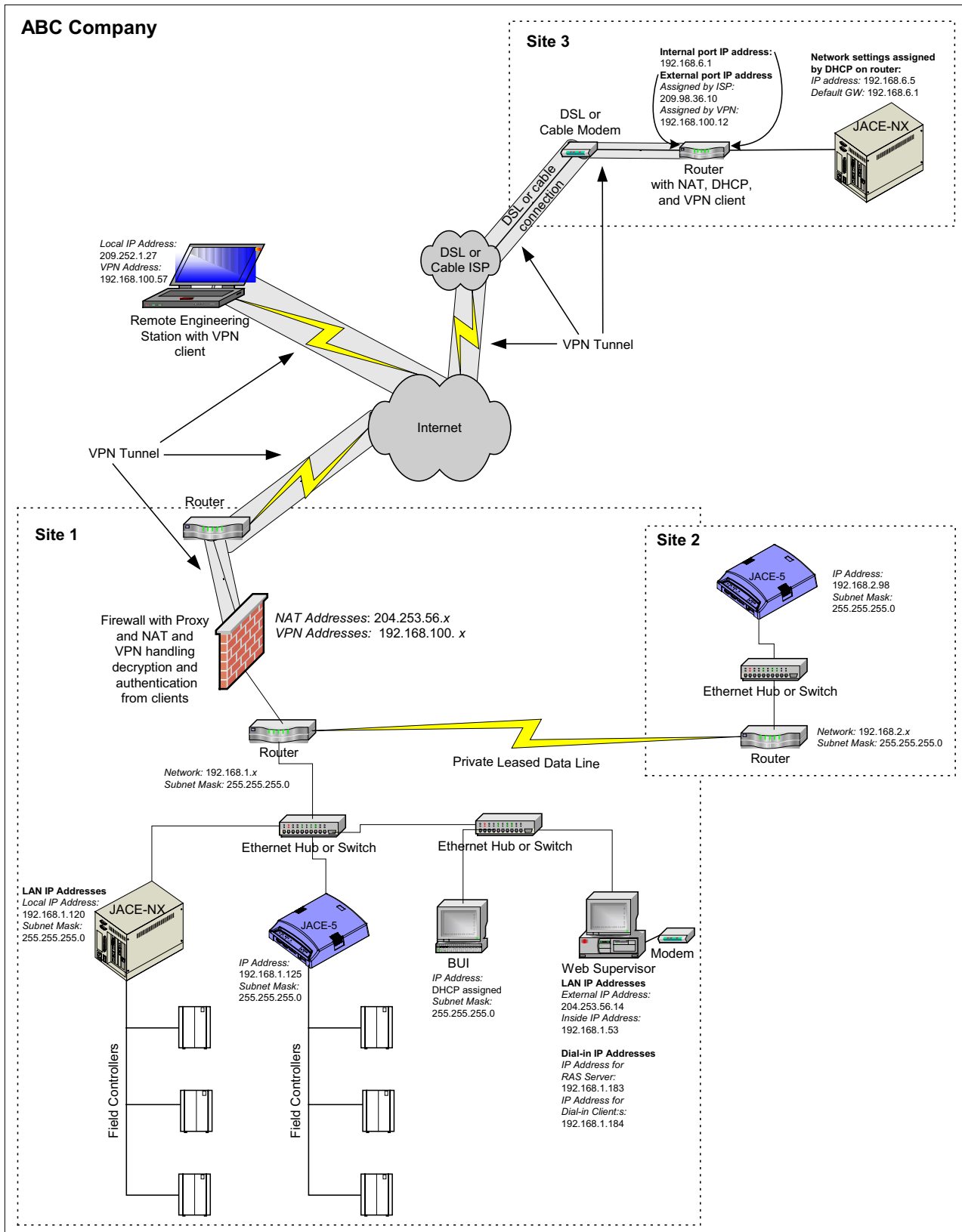
In this example, Company ABC has implemented VPN server software on their firewall, and added a new site (site 3). The router in site 3 has VPN client software, which has been configured to provide a persistent VPN connection to the firewall in site 1. After the router connects to the Internet, the client software connects to the VPN at site 1, receiving new network settings as defined by the VPN server. The router (and by extension, the JACE-NX) are now part of the LAN.

The company has also loaded and configured client software on the remote engineering station to allow the off-site SI to maintain NiagaraAX stations and hosts. Formerly, this maintenance was handled through dialup to the JACE-NX in site 1.

The engineering station connects to the Internet through its ISP, then initiates the VPN client. The client connects to the company firewall and the engineering host receives an IP address belonging to ABC company, joining its network. Until the

remote host disconnects the client software, all packets from the engineering host are routed onto the ABC company's network. The firewall has been configured to allow the remote engineering station access only to the NiagaraAX hosts available on the company network, including those in sites 1, 2, and 3.

Figure 7 Example VPN architecture



You should note the following things about using NiagaraAX hosts with a VPN:

- You cannot use a VPN with a QNX-based JACE connected directly to an ISP. That is because you cannot load VPN client software on a QNX-based JACE. You can use a QNX-based JACE with a VPN if the JACE connects to the Internet through an on-site router that provides VPN services (as well as DHCP and NAT). This is similar to the setup shown in site 3.
- This scenario has not been tested by Systems Engineering. We recommend that you set up a pilot to test them before implementing in a live job.
- Exact details on how to connect NiagaraAX hosts using VPNs cannot be provided due the many differences in VPN connection devices.

About NiagaraAX ports

As with most Internet-enabled applications, the applications running on NiagaraAX hosts also use default ports for communication with clients (typically other NiagaraAX hosts, or BUI users). For instance:

- a browser client that connects to a Px Page does so on port 80 of a NiagaraAX host running WebUI services (the port is configurable in the station's WebService properties).
- an engineering PC (the client) uses workbench to connect to a station for maintenance, by default Fox port is 1911.
- an engineering PC (the client) uses the Platform Tool to change an IP address on the host, and connects on the host's daemon port 3011.

Table 1 provides a list of the types of communication used by NiagaraAX hosts, and the default server ports used. Unless otherwise noted:

- the client randomly chooses an available client-side port (1024 or greater) to talk to the listed server port
- the server port listed is a TCP port (rather than a UDP port)
- most of the ports in the table are constantly open. Some applications (such as FTP, or the NiagaraAX web server) may constantly keep a port open, which means the port is scannable by a port scanner. Other ports (such as the host Admin port) only are shown when they are in use.

Table 1 TCP/IP ports used in NiagaraAX, various drivers, features, and functions

| Function | TCP Port | UDP Port | IP Protocols | Notes |
|--|---------------|---------------|--------------|--|
| NiagaraAX | | | | |
| Fox Service (Workbench) | 1911 | — | Fox, HTTP | Default port for a station's FoxService Used for Workbench-to-station and also station-to-station communications |
| Web Service | 80 | — | HTTP | Default port for a station's Web Service. Used in browser access (Hx, WB) |
| Encrypted Web Service | 443 | — | HTTPS | Default port for a station's Secure Web Service. Used in browser access (Hx, WB) |
| Platform daemon (access of) | 80, 3011 | — | HTTP | Default ports for WB platform connection. |
| Optional Functions | | | | |
| FTP | 21 | — | — | |
| FTP-data | 20 | — | — | |
| Telnet | 23 | — | — | |
| Client connection to mail server for e-mail notifications | 25 | — | SMTP | In NiagaraAX or r2.3.5 and later, the MailService lets you specify a TCP port other than 25 (default). |
| Microsoft Remote Desktop | 3389 | — | — | JACE NX with Win XP (Full or Emb) only |
| Internet Time Protocol service | 37 | — | — | |
| DHCP | — | 67, 68 | — | |
| NiagaraAX Drivers * | | | | |
| *Theses are "conventional" ports used by this particular protocol and driver. It is possible that another that another port is used on a particular job. If so, this other port must be unlocked, and also specified within the configuration of the corresponding NiagaraAX driver. | | | | |
| SNMP | — | 161 | SNMP | SNMP protocol |
| SNMP Trap | — | 162 | SNMP | |
| Modbus TCP | 502 | — | — | |
| BACnet Ethernet | — | — | — | TCP/IP not used (raw Ethernet) |
| BACnet/IP | — | 47808 | — | |
| BmsAdc | 25020 | — | — | |
| AreaDAT | — | 20028 | — | |
| CCN | — | 50005, 50006 | — | ComfortWorks Tunneling only |
| OPC Client (uses DCOM) | 135 | 135 | — | DCOM, using RPC (below). Filtering is hard because of dynamically assigned ports. |
| Others | | | | |
| RPC, used by NetBIOS (browsing, file shares, etc.) also Windows Update, Browser, OPC client & server | 137, 138, 139 | 137, 138, 139 | — | Required if JACE to appear in browser lists and for network shares. Used to copy future update files. Required by OPC client driver too. |
| PING | — | — | ICMP | Basic "ping" test of connection. |
| DCOM | 135 | 135 | — | See notes above for OPC Client driver. |
| Microsoft SQL Server | 1433 | 1433 | — | If R2 MS SQL or MSDE database option. |
| Microsoft SQL Monitor | 1434 | 1434 | — | SQL management only |
| Win XP Internet Time function | — | 123 | — | Win XP only, sync with NIST time server. |

CHAPTER 4

Network traffic considerations

Network traffic and bandwidth considerations are an important aspect of designing and implementing any NiagaraAX application. The purpose of this section is to provide information that can help you design a network architecture using a NiagaraAX application that is appropriate for your networking environment. There is an unlimited number of variations and many possible scenarios for a NiagaraAX networking environment. The following sections simply provide some information and guidelines that may help you understand how to design and manage network communications in order to optimize your NiagaraAX application. Each networking environment must be evaluated on an individual basis, monitored, and adjusted, as necessary to provide the optimum NiagaraAX networking environment.

The following sections address bandwidth and communication parameters that affect the overall efficiency of network communications. Use the information as a general guide only for designing and tuning the NiagaraAX network environment.

NiagaraAX network communications considerations

NiagaraAX communication may involve using different types of:

- programs or processes
- protocols
- platforms

These programs, protocols, and platforms are discussed in the [NiagaraAX architecture](#) section. All types of communication take place over the network at intervals that can be controlled to some degree by station and driver-specific “tuning” and “polling” controls in NiagaraAX. These communication parameters provide you with the ability to “loosen” or “tighten” the communication requirements that occur across a network. The total amount of information that exists as network “traffic” is largely dependent on how many points of information reside at host stations (in terms of points, alarms, histories, schedules, and more), how many clients are sharing that information, and how often it is updated. In order to effectively manage network communications, you need to understand where message and file transfers (downloading) exist and understand how often they occur.

The following sections provide some information that can be used as a guideline for gaining an understanding of how to control and adjust the amount of traffic that a NiagaraAX host and client may put on a network.

- [Types of browser communications](#)
- [Workbench-to-station communications](#)
- [Station-to-station communications](#)
- [Platform connection communications](#)

Types of browser communications

Browser communication can occur between any host that supports a standard web browser and any NiagaraAX host.

Traffic is created by both the initial page download and by page updates that occur using HTTP and Fox protocol. The size of each page in a browser is dependent on the individual page content, whether you are using web workbench or the Hx browser view.

HTTP communication HTTP protocol is used to download the workbench applet (`wbapplet`) and modules that are required for running workbench in the browser. Other HTTP traffic may be for graphics and widget downloads from various modules, as well as value updates when using Hx technology (non web workbench view). Each page and module download is cached so that they are a one-time traffic source, whereas the traffic involved with repetitive data value traffic is ongoing and dependent on the quantity and update frequency of information on a viewed page.

Fox communication Fox protocol communication occurs in a browser context only when you are using workbench in the browser (`wbapplet`). This web workbench tool handles communication between the host and client using the Fox protocol and port 1911. If you are using the Hx (non-applet) view, then value updates in the browser are made using Hx technology, via HTTP, and not the Fox protocol.

About the wbapplet

The workbench applet (`wbapplet.jar`) allows the full Niagara stack (or a subset of it) to download to a client machine using a web browser with the Java plug-in (required). Once loaded to the client, workbench runs as an applet within an HTML page. To create this full-featured workbench browser, the primary network traffic involved is the initial applet and module download. The download sequence is as follows:

- Based on the login-identity, an html file (see [Figure 4-1](#)) is opened in the browser.

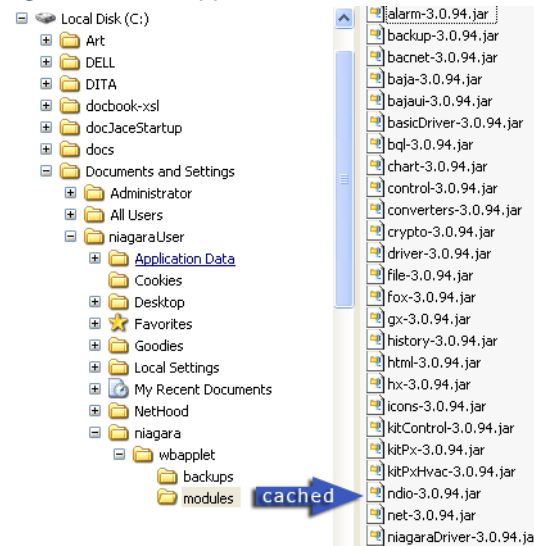
Figure 4-1 WbApplet call embedded in html file

```

<html><body style='margin: 0; padding: 0; bgcolor='#ffffff'>
<embed type='application/x-java-applet;version=1.4'
width='100%' height='100%' align='baseline'
pluginspage='http://java.sun.com/j2se/1.4/download.html'
boxbgcolor='#ffffff'
boxmessage='Loading Niagara...'
image=' '
codebase='/'
archive='wbapplet.jar'
code='WbApplet.class'
ord='station:|slot:/'
locale='en'
profile='workbench:DefaultWbWebProfile'
user='admin'
><noembed>
Java Plug-In support is required
</noembed>
</embed>
</body></html>

```

- The `wbapplet.jar` file downloads to the browser host and directs the required modules to download to the browser cache, as shown in [Figure 4-2](#).

Figure 4-2 WbApplet call embedded in html file

The total number of modules that are downloaded is variable, depending on the workbench profile that is downloaded. For example, if the host station does not have the weather service loaded, then the weather module is not downloaded. Additional modules may be downloaded and cached as you navigate to different pages in the browser. However, each module only needs to download once, since it is cached. Module file sizes vary from 2 kilobytes for the smaller ones to over a megabyte for some of the larger modules. You can see the individual file sizes under the NiagaraAX installation `modules` directory of your AXSupervisor workstation.

Workbench-to-station communications

Workbench-to-station communications (non-platform communications) occur when you view a remote station from workbench. Typically, this is during remote station engineering operations. Workbench-to-station traffic uses the Fox protocol and port 1911. Using the web workbench (embedded Workbench) both ports 80 and 1911 are used for HTTP access.

Remote programming can be a major consumer of both bandwidth and CPU resources. Proxy points in workbench represent source points in a station. Following, are some of the workbench features that enhance remote programming and can have a big influence on network traffic and CPU usage:

“Lazy loading” of proxy points. Proxy points are not “loaded” (created in the remote workbench) until they (or one of their child components) are actually viewed in workbench. When a remote station is initially opened in workbench, components are not loaded automatically, but they load when you view the component. This prevents a large initial surge in communication between the remote station and workbench. When a point is loaded, all its ancestors are loaded and they are both cached for the life of the workbench session. A loaded state does not mean that the component status and values are “current” with the master component in the remote station. To make sure that the component values are synchronized and up to date with the master, the proxy component must be in a “subscribed” state.

Subscriptions Subscription is a mechanism that allows proxy components to be notified when they need to be synchronized. This allows for savings in CPU, memory, and bandwidth resources because components that are not of current interest may be released from subscription (unsubscribed), thus saving resources. A subscribed state usually involves polling or eventing in order to maintain synchronization.

Note: *It is possible to chain subscriptions across multiple tiers—causing delays and stale information in the proxy point. For example, you could subscribe to a component in workbench that subscribes to the master in a station. In turn, that station component could be a proxy point for a piece of data that is running in a JACE. The JACE component could be modeling an external device, thus involving another polling operation. This could continue or be multiplied so that update delays are caused and network traffic increased.*

Nav tree synchronization NavEvents communicate across the network to maintain the proxy tree structure in a remote workbench view independent of the complete set of component information. This allows the proxy points to maintain their structure and identity in the nav tree without passing all component information every time the proxy component is changed or moved in the tree.

Leasing NiagaraAX provides a feature called “Leasing” that ensures that a proxy component is synchronized, but only as a snapshot for immediate use. A lease is a temporary subscription, typically for a minute, that reverts back to an unsubscribed state at the end of its lease time. This concept provides savings in resources by minimizing the time the subscribed state is maintained.

Batch calls It is very easy to create an undesirable situation with many round-robin network calls in your application environment. This can lead to excessive network traffic and result in poor application performance as well. It is almost always more efficient to use one large batch network call than many small network calls. NiagaraAX provides APIs to batch many common operations, including resolving multiple ORDs to a component or batching subscriptions. For more details about batching calls, refer to the *NiagaraAX Developer Guide*.

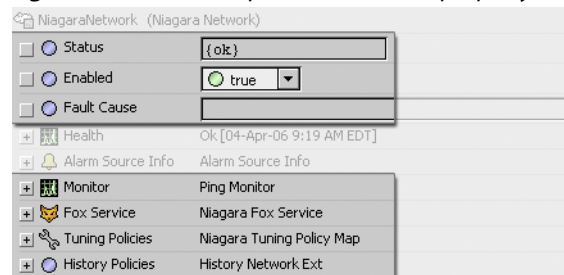
Station-to-station communications

Station to station communications occur over port 1911 using the Fox protocol and primarily involve messaging for point, alarms, histories, and schedules.

Station-to-station communications can include communications from remote stations to a supervisor station as well as peer-to-peer station communications. It is possible to have a large number of stations communicating with a supervisor or involved in peer communications, or very likely, both types of communication occurring. When you are working with many stations and especially when a large number of points, alarms, histories, or schedules are involved, it is critical to look at the overall communications architecture of your application. There are many possible communication scenarios that can create a large volume of message traffic on the network. It is important to understand how to use NiagaraAX features that allow you to control and schedule the generation and quantity of station-to-station traffic.

In the Network component property sheet there are properties that display in most drivers, as shown in [Figure 4-3](#).

Figure 4-3 Network parameters in the property sheet view



Status This is a read-only property that indicates if the driver is capable of communications. For any configured driver, network `Status` is typically `{ok}`. However, when adding some networks, the initial `Status` may be `{fault}`. This might mean a communications port is unassigned, or there is a port conflict.

A fault status also occurs if the host platform is not properly licensed for the driver being used. If you have a network fault, see the `Fault Cause` property value for more details. If you set the `Enabled` property to `false`, the status reads `{disabled}`.

Enabled By default, network `Enabled` is `true`--you can toggle this in the property sheet, or in the **Driver Manager** (by selecting the network and using the **Edit** button).

Whenever you set a network to disabled (`Enabled = false`), the network component and all child driver components (all devices, proxy points, etc.) change to a disabled status. If this is a field bus driver, communications routines like polling and device status pings also suspend. By default, disabled status color is gray text on a light gray background. This network-wide action may be useful for maintenance purposes to shut down communications across the driver network.

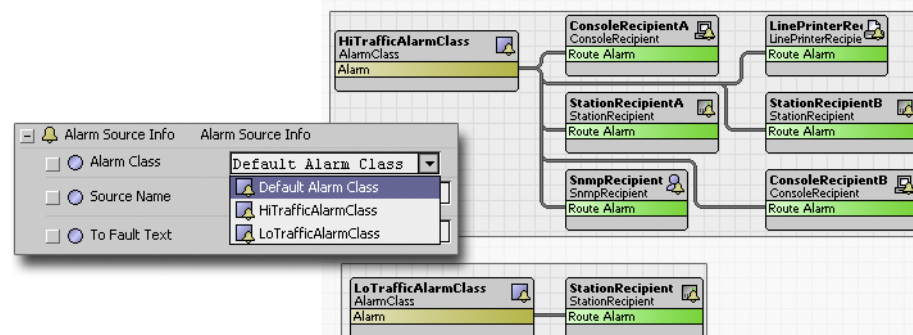
Note that `Enabled` is also individually available at the device-level and proxy point-level, using the **Edit** button or dialog box in the **Device Manager** or **Point Manager**, respectively. A disable at the device-level disables all child (proxy points), as well as polling to that points under that device. A disable at the proxy-point level disables the single point.

Health This property simply displays advisory information about the “health” of the particular driver network. These read-only properties can help you recognize and troubleshoot a network problem but they provide no direct network management controls.

Alarm Source Info A network's Alarm Source Info slot holds a number of common alarm properties that are used to populate an alarm if the network does not respond to a monitor ping. The `Alarm Class` option property may have impact on the network traffic since different alarm classes may be configured to impact network traffic to different degrees. For example, you may have an alarm class configured to route alarms simultaneously to multiple recipients across the network, including email, lineprinter and remote station recipients.

Each child device object also has its own Alarm Source Info slot, with identical (but independently maintained) properties.

Figure 4-4 Alarm class choice affects network traffic



Monitor Monitor provides verification of the general health of the network, plus the network's “pingables” (typically, devices) by ensuring that each device is minimally “pinged” at some repeating interval.

- **Ping Enabled** You should leave `Ping Enabled` set to `true` in almost all cases. In this state, a ping occurs for each device under the network, as needed. While set to `false`, device status pings do not occur and device status message displays cannot change.

- **Ping Frequency** Specifies the interval between periodic pings of all devices. Typical default value is every 5 minutes (05m 00s), you can adjust differently if needed.

The network ping Monitor will only “ping” the device if the time of last health verification is older than the ping frequency. Therefore, in normal operation with most drivers, the proxy point polling mechanism actually alleviates the need for the monitor ping, providing that the ping frequency is long enough. The default

setting of 5 minutes between “pings” should be good for most situations. If the ping frequency is set to some very low setting and you have a lot of devices on the network, the message traffic for pinging could be significantly higher.

- **Alarm On Failure** If this property is set to `true` (default), an alarm is recorded in the station's AlarmHistory upon each ping-detected device event (“down” or subsequent “up”). If `false`, device “down” and “up” events are not recorded in the station's AlarmHistory.
- **Startup Alarm Delay** Specifies the period a station must wait after restarting before device “down” or “up” alarms are generated. Applies only if the Monitor's property `Alarm On Failure` is `true`.

Following, is a list of some of the network-level control options that are common across many of the drive networks. Each one of these child components has properties that are adjustable and can have network-wide impact.

- [Fox service](#)
- [Network tuning policies](#)
- [Network polling policies](#)
- [Common control properties](#)
- [History policies](#)
- [History import and export](#)
- [Schedule import and export](#)
- [Types of alarm service communication controls](#)
- [Time sync service](#)
- [Email service](#)

Fox service

The NiagaraNetwork's `Fox Service` holds configuration properties for the local station's Fox settings. Some of these settings can affect traffic between the local station and other stations. Often in a multi-station job, in each station you create a user account specifically for station-to-station communications, with an assigned profile type of “super user.” This is the account that you reference when you edit a NiagaraStation's Client Connection properties, entering its username and password.

The Fox Service also has a `Server Connections` container slot with a default **ServerConnectionsSummary** view. Client connections to the station's Fox server are dynamically modeled as “SessionN” child components. The summary view allows you to see *all current connections*, and if necessary, perform a right-click **Force Disconnect** action on one or more connected users.

Fox Service properties are shown in [Figure 4-5](#) and completely described in the *User Guide*.

Figure 4-5 Fox service

The Fox service parameters that are more likely to significantly affect network traffic are mentioned below:

Max server connections This property limits the number of server connections that are allowed on the local station. When the maximum number of connections is reached, any successive attempts to logon to the station will fail. The default setting is 100 sessions. Adjust this parameter, as needed, to fit your requirements.

Trace settings The four `Trace` components allow you to return all message activity (verbose) on those parameters you set to `True`. This includes all transactional messages, which may result in too many messages to be useful. Increasing station output by assigning trace levels consumes extra station resources and exacts a performance penalty.

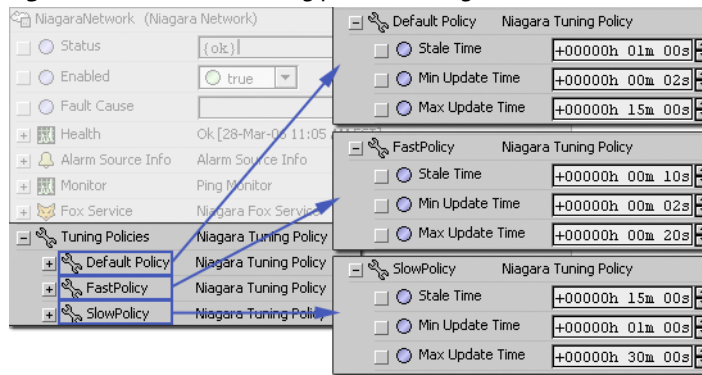
Note: *Be careful using Trace! After troubleshooting, remember to return log levels to default values.*

Network tuning policies

Tuning policies are important for providing network-wide controls for a particular network driver. [Figure 4-6](#) shows network level tuning options that are available from a driver property sheet view in workbench.

A network's Tuning Policies view holds one or more collections of "rules" for evaluating both write requests (e.g. to writable proxy points) as well as the acceptable "freshness" of read requests from polling. Some drivers also support different poll frequency groups (Slow, Normal, Fast) within their network tuning policies or under their poll scheduler component.

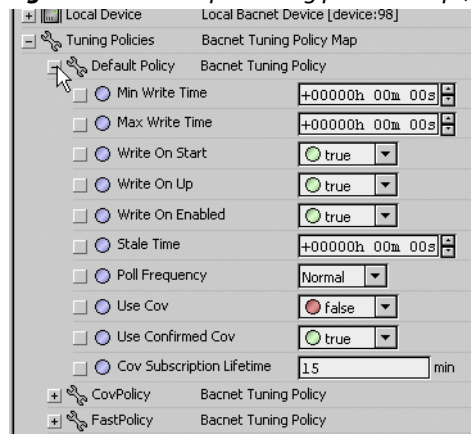
Figure 4-6 Network tuning policies – NiagaraNetwork



Note: Some driver networks do not have Tuning Policies. For example, an RdbmsNetwork for a database driver. Also, the NiagaraNetwork has greatly simplified Tuning Policies.

In the network’s property sheet, expand the Tuning Policies (Map) slot to see one or more contained Tuning Policies. Expand a Tuning Policy to see its configuration properties as shown in Figure 4-7.

Figure 4-7 Example tuning policies map (Bacnet)



By default, a driver’s Tuning Policy Map contains just a single TuningPolicy (“Default Policy”). However, you typically create *multiple* tuning policies, changing those items needed differently in each one. Then, when you create proxy points under a device in that network, you can assign each point (as needed) to the proper set of “rules” by associating it with a *specific* tuning policy.

As a simple example (under a BacnetNetwork), you could *duplicate* the default tuning policy *twice*, naming the first copy “Slow Policy” and the second copy “Fast Policy.” Then, in each copy change the “Poll Frequency” property from “Normal” to “Slow” or “Fast,” respectively. You would then have 3 available (and different) poll frequency groups to pick from when you create and edit proxy points.

Tuning Policy properties

Tuning Policy properties for typical field bus drivers are as follows:

Min Write Time Applies to writable proxy points, especially ones that have one or more linked inputs. Specifies the minimum amount of time allowed between writes so that it provides a method to throttle rapidly changing value so that only the latest value is written. If the property value is 0 (default), this rule is disabled (all value changes attempt to write). In this case, if you have a rapidly changing value and set a zero value for min write time, this single point can create frequent message traffic.

Max Write Time Applies to writable proxy points. Specifies the maximum “wait time” before *rewriting* the value, in case nothing else has triggered a write. Any write action resets this timer. If the property value is 0 (default), this rule is disabled (no timed rewrites). Larger values in this property *allow for* less message traffic by not *forcing* a write but they do not cause the write to occur.

Write On Start Applies to writable proxy points. Determines behavior at *station startup*.

- If true, (default) a write occurs when the station first reaches “steady state.”
- If set to false, a write does not occur when the station reaches “steady state.”

Write On Up Applies to writable proxy points. Determines behavior when proxy point (and parent device) transitions from “down” to “up.”

- If true, (default) a write occurs when the parent device transitions from down to up.
- If set to false, a write does not occur when the parent device transitions from down to up.

Write On Enabled Applies to writable proxy points. Determines behavior when a proxy point’s status transitions from “disabled” to normal (enabled). Note this status can be inherited globally if the parent device was set to disabled (or *network-wide* if driver network was set to disabled).

- If true, (default) a write occurs when writable point transitions from disabled.
- If set to false, a write does not occur when writable point transitions from disabled.

Stale Time Applies to all proxy points.

- If set to a non-zero value, points become “stale” (status stale) if the configured time elapses without a successful read, indicated by Read Status “ok.”
- If set to zero (default), the stale timer is disabled, and points become stale immediately when unsubscribed.

Note: *By default, proxy point status “stale” is indicated by tan background color. In addition, stale status is considered “invalid” for any downstream-linked control logic.*

Poll Frequency In the BacnetNetwork (only) TuningPolicy. Applies to all proxy points. Provides a method to associate the tuning policy with one of 3 Poll Rates available in the network’s Poll Service: Fast Rate, Normal Rate, or Slow Rate. The default poll frequency is “Normal.”

Depending on the driver, there may be a single “Poll Service” (or “Poll Scheduler”) slot under the network, or as in the case of a BacnetNetwork, a separate “Poll Service” for each configured port (IP, Ethernet, Mstp) under its BacnetComm > Network container. The NiagaraNetwork uses subscriptions instead of polling.

Cov Subscription Lifetime In the BacnetNetwork (only) TuningPolicy. Applies to all proxy points under this driver. Provides a method to limit the time that a Cov will maintain a Subscription state. The default time is 15 minutes.

Note: *Depending on the driver, there may be a single “Poll Service” (or “Poll Scheduler”) slot under the network, or as in the case of a BacnetNetwork, a separate “Poll Service” for each configured port (IP, Ethernet, Mstp) under its BacnetComm > Network container. The NiagaraNetwork uses subscriptions instead of polling.*

Tuning policies used by a specific driver may have unique characteristics. For example, under a NiagaraNetwork, TuningPolicy has only two properties: `Min Update Time` and `Max Update Time`.

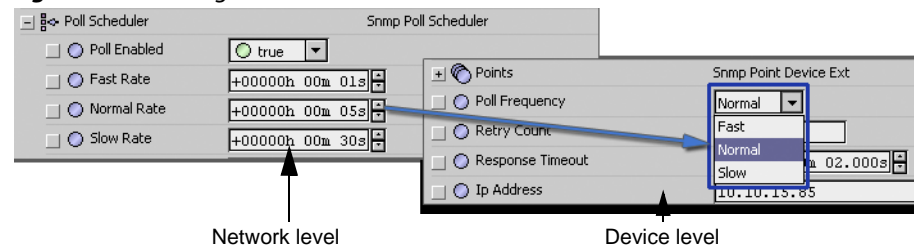
Other drivers may have specific considerations for tuning policies. For more information, refer to the “Tuning Policy Notes” section within any NiagaraAX driver document.

Network polling policies

Some driver networks use a single polling mechanism (a poll component, named “Poll Service” or “Poll Scheduler”) adjusted at the *network level* in the station. Regardless of driver type, the Poll Service provides a “poll scheduler” that scans four polling-rate categories to service pollable items, as follows:

- The Poll Scheduler allocates three rate categories (for example: Slow, Normal, Fast). Pollable items (mainly proxy points) are associated with one of these three rate groups. The rates are assigned by choosing a Tuning Policy in each point's proxy extension (BacnetNetwork and SnmpNetwork) or by using a Poll Frequency setting (drivers other than BacNet and Snmp).
- A fourth “dibs stack” category is allocated for pollable items that transition to a subscribed state. This may be a “temporary subscription,” such as results when viewing unlinked proxy points (without alarm or history extensions) in workbench or a browser. Or, it may occur when a permanently subscribed point is first polled (thereafter, it is no longer “dibs stack-polled”).

Figure 4-8 Polling scheduler



For more information about polling and the priority of polling by the scheduler, refer to the *NiagaraAX User Guide*.

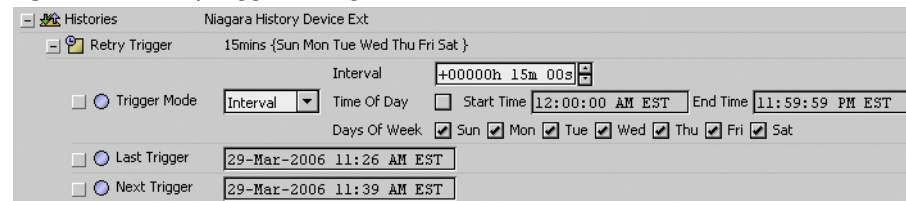
Common control properties

Several network control parameters are common to Schedule, History, and other network features provided under station to station communications. The following descriptions may apply to Niagara Network and other drivers.

Enabled An `Enabled` property is available at the network level, device (or station) level, and as part of the schedule and history import and export extensions. Setting `Enabled` to `true` allows the network, device, or component to operate and communicate. Changing the setting to `false` disables the network device, or component functions, thereby stopping any related network traffic that they might initiate.

Retry Trigger By default, device extensions Histories and Schedules contain a “Retry Trigger” component, appearing in the property sheet view but not in any Manager view. The retry trigger can have a significant impact on traffic if there are one or more unsuccessful import or export actions attempted.

Figure 4-9 *Retry trigger settings*



Upon any *unsuccessful* execution of a history import or export descriptor or schedule import or export descriptor contained in that same device extension, the Retry Trigger defines subsequent “retries” that will automatically occur upon the interval period defined (default value is 15 minutes). This continues until successful execution occurs. Obviously, a smaller time interval creates more “retry” traffic when communication between the remote and supervisor station is interrupted for an extended period.

Execution Time By default, import and export descriptors (associated with both Histories and Schedules) contain the `Execution Time` component. This component provides for flexible timing for executing an import or an export. Options for `Execution Time` settings are described under the “[Trigger Mode](#)” topic.

Figure 4-10 Execution time

If an Execution Time trigger fails, the Retry Trigger settings are invoked. The difference between the Retry Trigger and Execution Time component is that the Execution Time component fires at the defined “Execution Time(s)” whereas the Retry Trigger fires whenever a subordinate execution fails. It is possible to set an Execution Time to an interval as low as 1 second, thus creating constant import or export traffic. If an unusually low setting (frequent firing) is multiplied over many histories and calendars, it can be a source of excessive network traffic.

Trigger Mode This property allows you to choose a triggering mode for a “Retry Trigger” or an “Execution Time”. Trigger mode options are shown in [Figure 4-11](#) and described in the following list:

Figure 4-11 Trigger modes (shown in Execution Time component)

- **Manual:** requires a manual execution action in order to perform the import. No import takes place until a user takes an action to execute it.
- **Daily:** allows you to specify a time for import to occur on a once-daily basis. You may consider scheduling imports at different times across a network—possibly staggering the times if large history files are anticipated.
- **Interval:** allows you to specify a frequency and time period when imports occur.

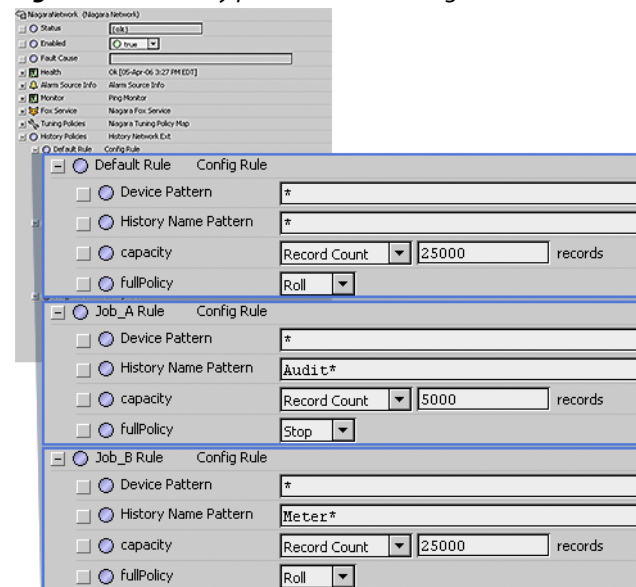
This option allows you the most flexible control options, but also allows you to increase network traffic significantly by decreasing the time interval between imports. It is possible to decrease the interval to as little as 1 second between imports, which is excessive for most applications.

History policies

The NiagaraNetwork's **History Policies** (`HistoryNetworkExt`) holds rules that are used when remote histories are “exported” into the local station. These are usually histories that are pushed from a remote station for archiving onto a supervisor station. Unlike *imported* histories, which let you define and adjust the `capacity` and `full policy` settings in each `HistoryImportDescriptor`, histories that are exported from one station into another station have no associated self-limiting parameters available at the exporting station. The `capacity` and `full policy` for each history is set at creation time, using the **local history policies** of the target station. These parameters affect the size of the history logs that are archived over the network. Usually the supervisor station is set to archive at a higher capacity than a remote station.

Figure 4-12 shows three different history policies under a NiagaraNetwork. These rules are all applied, in sequential order. Refer to the *User Guide* for more details about how the history policy rules are applied.

Figure 4-12 History policies under a Niagara Network



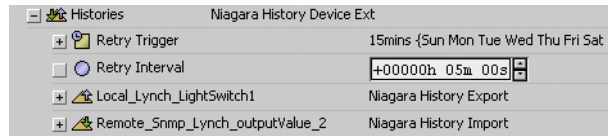
History import and export

Under the **NiagaraNetwork** and the **BacnetNetwork**, local data logging provides the ability to use the `HistoryDeviceExt`. This device-level extension serves as the parent container for history descriptors--components that specify how history-type collections (log data) are imported or exported. The `HistoryDeviceExt` property sheet, shown in Figure 4-13, has the following properties that you can use to adjust history-related communications:

Retry Trigger Refer to “[Retry Trigger](#),” page 4-14 for details.

Retry Interval The retry interval property also includes the ability to schedule “retries” at specific times or manually. This allows you to consider when and how often to allow import and export attempts across the network.

Figure 4-13 HistoryDeviceExt

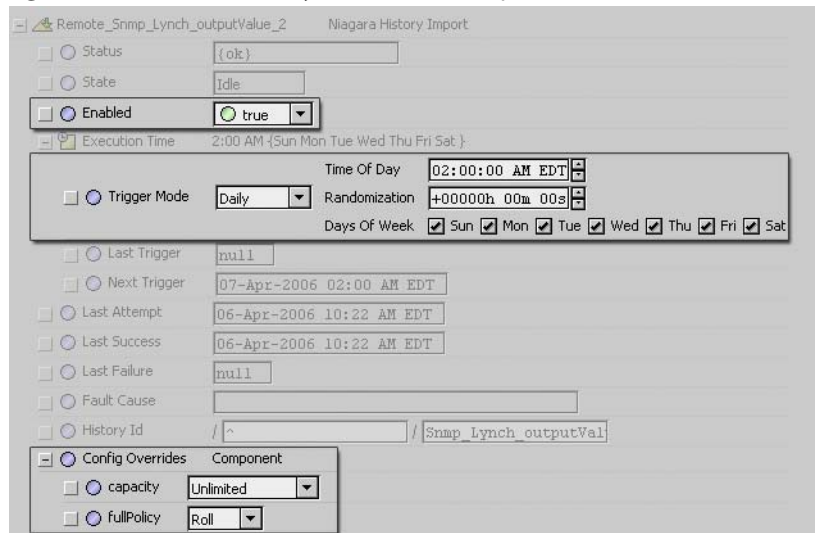


Creating history import and history export descriptors is how you save a history to a different location (station) from where it originated. In a typical application, this is considered “archiving.” For example, an originating history (with a limited record count) may be in a JACE station. If imported to a supervisor station, its history import descriptor can be configured such that the imported history in the supervisor has “unlimited” record capacity. The JACE history can run collecting only the last 500 records, while the imported history in the supervisor will collect all (unlimited) records. These settings affect the size of the history logs that are archived over the network.

History import

History import provides some controls that allow you to limit or expand the size of the files that are archived to the local station. These properties are shown in [Figure 4-14](#) and described as follows:

Figure 4-14 Selected history device extension parameters



Enabled Refer to “[Common control properties](#)”, for details.

Trigger Mode Refer to “[Common control properties](#)”, for details.

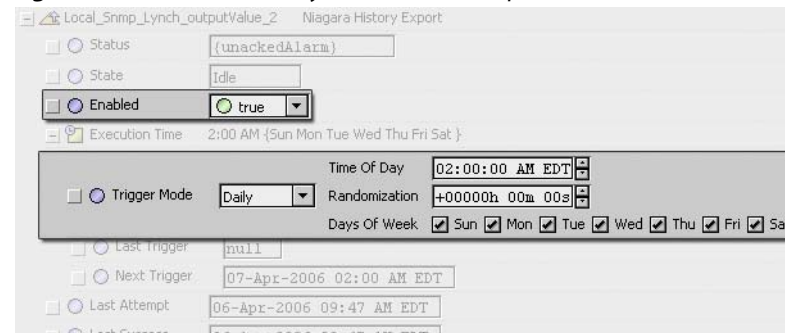
Config Overrides: This component allows you to specify a number for `Capacity` and `Full Policy`. These parameters allow you to control the number of records that you pull in on an import (`Capacity`) and (if a capacity number is set) to specify what to do when that number is reached (`Full Policy`).

These “override” options provide a way to limit import file size and bandwidth and are called “overrides” because they take priority over the values that are specified in the remote station under a point’s individual **History Ext**, **History Config** properties.

History export

History export provides controls that allow you archive history files at defined times and intervals. To limit or expand the size of the files that are archived to the local station. These properties are shown in [Figure 4-15](#) and described as follows:

Figure 4-15 Selected history device extension parameters



Enabled Refer to “[Common control properties](#)”, for details.

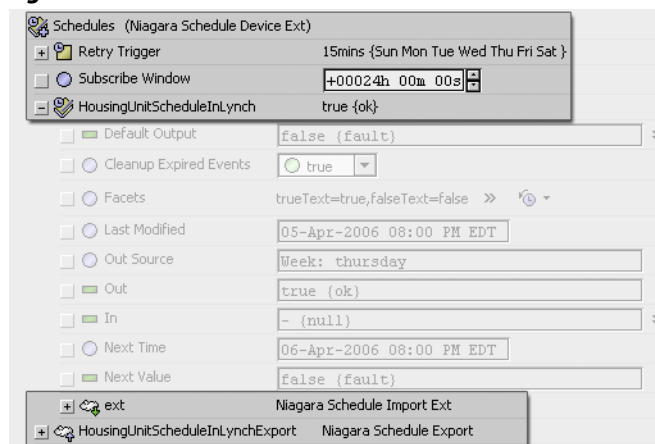
Trigger Mode Refer to “[Common control properties](#)”, for details.

Schedule import and export

You can import or export schedules under **NiagaraNetwork** and **BacnetNetwork** devices. Typically, schedules on the “import” side have “Execution Times” (time triggers) that are configured to execute manually, while those that are on the “export” side are configured to export (or push) on a daily basis or on a recurring interval.

Under the **NiagaraNetwork** and the **BacnetNetwork**, local schedules provide the ability to use the network schedule device extension. This device-level extension serves as the parent container for schedules and associated schedule import and export descriptors--components that specify how schedules are imported or exported. This also contains a Retry Trigger, see “[Retry Trigger](#),” page 4-14. The `HistoryDeviceExt` property sheet is shown in [Figure 4-13](#).

[Figure 4-16](#) shows the options available for setting update method and intervals.

Figure 4-16 Schedule device extension

To avoid unnecessary network traffic and use of CPU resources, it is important to be aware of the execution frequency and fixed execution times so that you avoid unnecessarily short execution times and duplicated (or simultaneous) network-level updates.

Schedule import and history export descriptors allow you to save a schedule to a location (station) different from where it originated. In addition to the individual import and export descriptors, this component has the following controls that can affect your network traffic.

Retry Trigger Refer to [“Common control properties”](#), for details.

Subscribe Window At a random time after station startup and within the time specified by the SubscribeWindow property value, all subordinate schedules who have not communicated with their supervisor will have their execute action invoked. For drivers where remote supervisors do not persist information about local subordinates, the subscribe window should be some small value rather than the default of a day. If you have many subordinate schedules, a larger Subscribe Window time decreases the chance of simultaneous schedule import or export execution.

Types of alarm service communication controls

The alarm service has the certain types of controls that can affect network traffic levels. Most of these parameters work so that they either allow or disallow communication at certain times

Figure 4-17 Common alarm service controls

The screenshot shows the 'StationRecipient' configuration window. The 'Time Range' section is expanded, showing 'Start Time' and 'End Time' both set to '12:00:00 AM EDT'. The 'Days Of Week' section has checkboxes for Sun, Mon, Tue, Wed, Thu, Fri, and Sat, all of which are checked. The 'Transitions' section has checkboxes for 'toOffnormal', 'toFault', and 'toAlert', all of which are checked. The 'Default Alarm Class (Alarm Class)' section is also expanded, showing 'Ack Required' checked, 'Priority' set to 'toOffnormal 255 toFault 255 toNormal 255 toAlert 255', 'Total Alarm Count' set to '60', 'Open Alarm Count' set to '1', 'In Alarm Count' set to '0', 'Unacked Alarm Count' set to '1', and 'Time Of Last Alarm' set to '23-May-2005 11:37 AM EDT'.

Ack required This control gives you the option of selecting or clearing the alarm acknowledgement requirement associated with the alarm generation conditions in an alarm class or an alarm recipient, as shown in [Figure 4-17](#).

If alarm acknowledgement is set (box is checked) for any one of the following alarm types, an alarm acknowledgement is routed back to the alarm source when that alarm is acknowledged—thus creating network traffic.

Time range The `Time Range` parameter applies to several components related to the alarm service. Setting the time range allows you to limit when communications can occur using the parent component. For example, an alarm class or alarm recipient type can only communicate during the time set in the `Time Range` window. [Figure 4-17](#) shows the `Time Range` component.

Days of week The `Days of Week` parameter allows you to limit the alarm communications to certain days of the week.

Remote station This parameter allows you to route alarms to a remote station. Alarms that are sent to a remote station are routed according to the parameters defined in the Alarm class that is selected in the Alarm component under the remote station's `NiagaraNetwork`.

Rout acks This parameter allows you to enable alarm routing acknowledgments (`true`) or disable alarm acknowledgments (`false`).

Time sync service

The `TimeSyncService` has certain parameters that you can adjust to increase or decrease the frequency of its polling. This includes the following:

Client Schedule (Time Trigger) This control works the same as other time triggers. Refer to [“Common control properties,”](#) page 4-14 for a description of the properties associated with the `TimeTrigger` component.

TimeSyncClient This control has 2 parameters that allows you to set the number of retries for a client (`Retries`) and the time for a client connection to time out (`Timeout`). The default numbers 5 (for retries) and 250ms (for timeout) should be sufficient for most applications. If a relatively large number is used here, unnecessary traffic could be generated with unnecessary retries.

Email service

The email service has an outgoing account component that you can configure. The following properties may affect traffic on the network:

Pollrate This property controls how often the host is polled. Increasing the `pollrate` value increases the time between polls. During the time between polls, emails may be queued (up to the max queue size) until the next poll time. At the next poll time all queued emails are sent.

Max queue size This is the maximum number of emails that are allowed to be in the queue. This allows you to limit the number of emails that can be queued, so that there is a limited amount of email traffic when polling is enabled and the queued emails are sent. If you have a very high number set here, a large amount of email traffic could be generated when email is enabled.

Max sendable per day This property allows you to limit the number of emails that are sent in one day.













Platform connection communications

A platform connection is different than a station connection. When connected to a Niagara platform, Workbench communicates (as a client) to that host's platform daemon (also known as "niagarad" for Niagara daemon), a server process. Unlike a station connection, which uses the Fox protocol, a client platform connection requires workbench, meaning it is unavailable using a standard Web browser. The default ports for platform communications are ports 3011 and port 80.

Communication at a platform level may occur when a platform connection exists via the platform daemon. Maintaining a simple view of the platform director does not create any regular traffic, however, any views that have updates create update traffic proportionate to the number of updating fields that are on the view.

Figure 4-18 shows the platform view. Most of the functions available from this view create traffic and bandwidth usage directly related to the number and size of files involved. For file transfers (including modules) you can look at the file sizes before making the changes to assess how much traffic will be hitting the network and to decide if the target JACE (for example) has enough storage space to handle the transfer. Several platform functions cause insignificant amount of network traffic.

Figure 4-18 Platform view

| Platform | |
|---|--|
| Name | Description |
|  Dialup Configuration | Configure the way the remote host uses dialup networking |
|  Distribution File Installer | Install distribution files to the remote host |
|  File Transfer Client | Transfer files to and from the remote host |
|  Lexicon Installer | Install lexicons to support additional languages |
|  License Manager | Manage licenses and certificates |
|  Platform Administration | Update the platform daemon's port or credentials, or set its date and time |
|  Software Manager | Install software to the remote host |
|  Station Copier | Transfer stations to and from the remote host |
|  Station Director | Control stations and access console output |
|  TCP/IP Configuration | Manage the host's TCP/IP settings |
|  User Manager | Manage the local OS users and groups |
|  Remote File System | The remote host's file system |

Note: *The provisioning service has the potential to put a greater amount of traffic on the network due to the ability to target multiple platforms.*