Technical Document

Niagara 4 Networking and IT Guide

October 7, 2019

Niagara 4 Networking and IT Guide

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

Confidentiality

Trademark notice

Copyright and patent notice

Contents

About this Guide

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

Product Documentation

This document is part of the Niagara technical documentation library. The information in this document is written primarily for IT professionals.

Document Content

This guide provides basic information about Niagara Framework architecture, platforms, protocols and standard networking technologies. Topics included describe how to integrate Niagara devices into a network environment and implement networking security strategies to help corporate IT managers or IT personnel get started. The document also provides information about how to optimize the performance in complex systems.

Document change log

Updates to this document are listed below.

October 7, 2019

In the topic, "Security best practices", added a caution note alerting customers to restrict access to all computers, devices, field buses, components, etc., that manage their building model.

August 22, 2017: Updates

- Added to the fourth bullet in "Security best practices".
- Rewrote occurrences of "is connected to the Internet" to read, "is exposed to the Internet."
- Updated the VPN connection represented in the graphics found in these topics: Single-site network application, page 21 and Multi-site network application, page 22.

August 10, 2017: Initial release publication

Related documentation

Additional information on Niagara system, devices and protocols is available in the following documents.

- *Getting Started with Niagara*
- *Station Security Guide*

Chapter 1 Architecture

Topics covered in this chapter

- ♦ Java
- ♦ Heterogeneous system integration
- ♦ Component software
- ♦ Platforms
- ♦ Programs
- ♦ Communication protocols

The Niagara Framework® is a Java software framework for integrating disparate building automation systems into a single, manageable interface that can run on multiple hardware platforms.

Java

The framework uses the Java Virtual Machine (JVM) 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 Java 8 SE (Standard Edition) compact3 profile compliant VMs (Virtual Machines). The user interface toolkit and graphical programming tools are targeted for Java 8 SE VMs.

Heterogeneous system integration

Niagara is designed from the ground up to assume that there will never be any one standard network protocol, distributed architecture, or fieldbus. Instead, the framework integrates cleanly with all networks and protocols, standardizing the contents of the box, not what the box talks to.

The framework is targeted for embedded systems capable of running a Java VM. This excludes low-end devices without 32-bit processors and several megabytes of RAM. But, even embedded systems with the power of low-end workstations have special needs. They are always headless 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.

The framework also scales to highly distributed systems composed of tens of thousands of nodes running the framework software. Systems of this size span a wide range of network topologies and usually communicate over unreliable Internet connections. The framework is designed to provide an infrastructure for managing systems of this scale.

Component software

Framework architecture is centered around component-oriented development. Components are pieces of self-describing software that can be assembled like building blocks to create new applications.

This component-centric architecture solves many problems:

- Components normalize the data and features of heterogeneous protocols and networks so that they can be integrated seamlessly.
- Components along with the graphical tools provided by the framework allow applications to be assembled without requiring a Java developer.
- Components provide unsurpassed visibility into applications. Since they are self-describing, tools can easily interrogate how an application is assembled, configured, and what is occurring at any point in time. This aids debugging and application maintenance.

- Components enable software reuse.

A set of Java APIs serve dual purposes: Developers can access them to write Java code, while non-programmers can use high-level graphical programming and configuration tools to create custom applications. This vastly increases the scope of users capable of building custom applications.

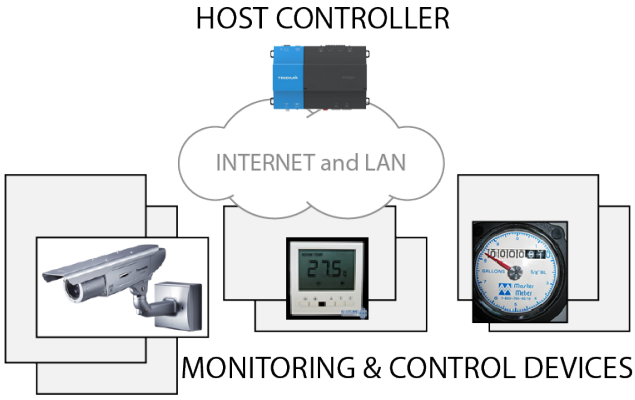
Platforms

The framework is hosted on a wide range of platforms, from small embedded controllers to high-end servers.

JACE controllers

The simplest configuration consists of a single JACE (host) controller.

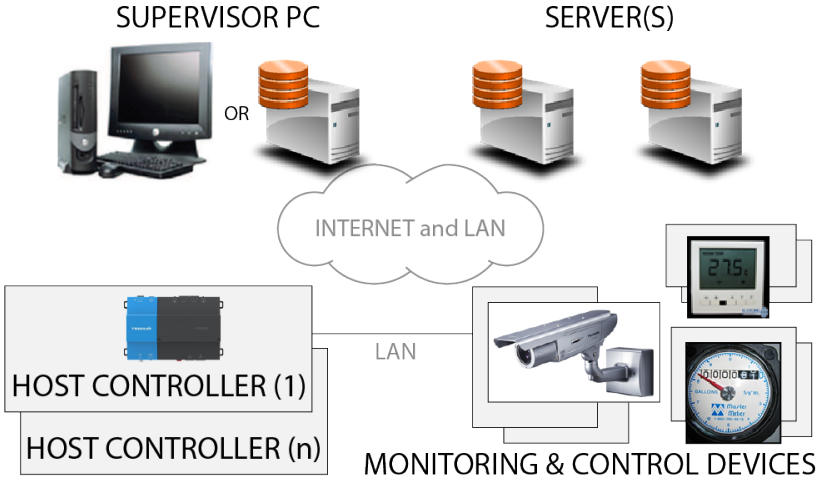
Figure 1 Single JACE controller configuration



This dedicated host platform provides integrated control, supervision, and network management services for a network of building monitoring and control devices.

Multiple host controllers may be combined in a distributed network that includes a Supervisor PC (laptop or desktop) to provide a variety of control and reporting functions.

Figure 2 Supervisor with multiple JACE controllers configuration



Controllers host a station and a daemon process, but not Workbench. They typically run QNX as their operating system.

Supervisor

A computer, workstation or server-class machine connected to the network is called a Supervisor. This device is a network PC that acts as a server for multiple controllers. A station runs in the Supervisor computer. The daemon and Workbench may also run in the Supervisor computer.

A Supervisor running Workbench has the tools to:

- Provision (install and update) software modules on multi-controller systems
- Integrate support for standard RDBMS (MS SQL, Oracle, Mysql, Hsql, etc.)
- Serve as a platform for enterprise applications
- Provide central database storage for the attached controllers
- Serve as an archive destination / repository for log and alarm data
- Serve as a central server (a single IP address) for delivering graphics and aggregated data

Programs

There are typically four different programs (or processes) associated with a Niagara Framework system: a station, the Workbench tool, a daemon process, and a web browser.

Station

An instance of the framework running on a platform is called a station. A station runs the components of the framework and provides access for client browsers to view and control the components. The primary parts of a station include components and services. The station can run on a PC or the JACE controller.

Workbench

Workbench

Workbench is an engineering tool, a Java VM, that manages Niagara components. It provides a means to accomplish platform tasks, such as:

- Launching and monitoring stations
- Installing and backing up station databases
- Configuring TCP I/P settings
- Installing and upgrading the platform OS (QNX only), Java VM, and Niagara software
- Installing software licenses

Daemon

The daemon is a native process used to boot stations and to manage platform configuration, such as IP settings. On Windows platforms, the daemon runs in the background as a Window's service. On QNX platforms it runs as a daemon process on startup. The most common way to access daemon functionality is through Workbench.

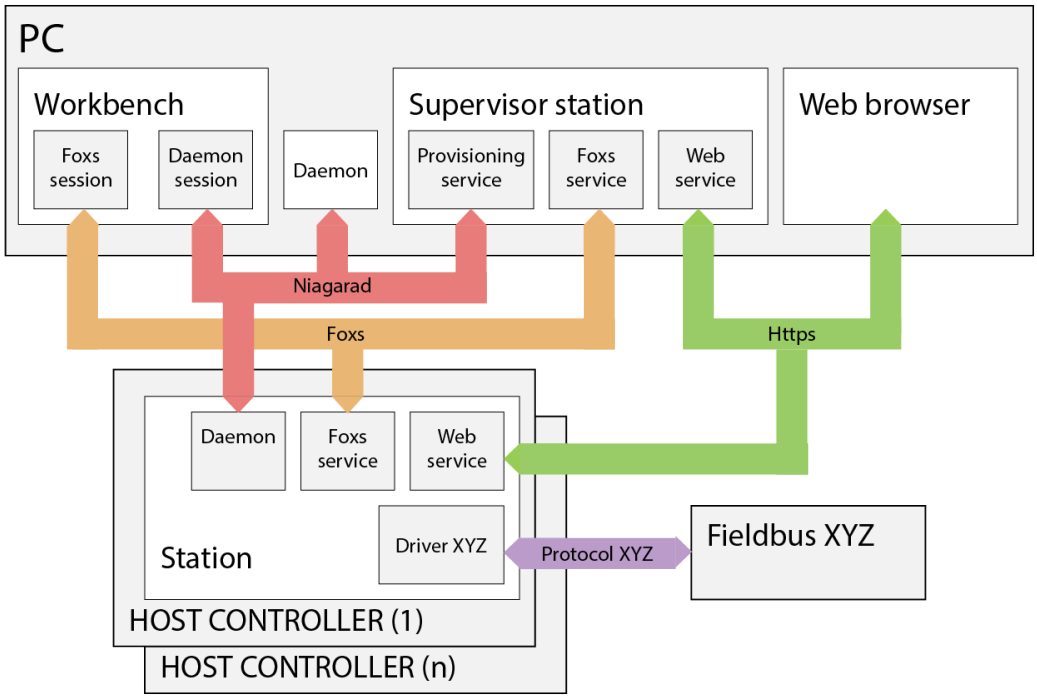
Web browser

A standard web browser such as IE or Chrome hosts one of Niagara's web user interfaces. Niagara provides both server-side and client-side technologies to build web UIs.

Communication protocols

Three network protocols integrate the four programs (station, Workbench, daemon and web browser). One or more additional protocols (drivers) communicate with specific devices (Driver XYZ, Protocol XYZ and Fieldbus XYZ).

Figure 3 Communication protocols using JACE hosts



- Fox/Foxs is a proprietary TCP/IP protocol used for station-to-station and Workbench-to-station communication. The Fox Service in each station defines the port to use and manages access.
- HTTP/HTTPs is a standard protocol used by web browsers to access station web pages. The Web Service facilitates communication over HTTP.
- Niagarad/platformtls is a proprietary protocol used for Workbench-to-daemon communication. In the Supervisor station, Niagarad also communicates with the Provisioning Service. This service automates the performing of tasks on remote controllers.

Encrypted versions of these protocols: Foxs (Fox-secure), HTTPs (HTTP-secure), and platformtls (secure niagarad) are the preferred configuration defaults.

Chapter 2 Security best practices

Topics covered in this chapter

- ♦ Security
- ♦ Threats and risk assessment
- ♦ Framework security features
- ♦ Code-signed modules
- ♦ Data access control
- ♦ Multiple Credentials
- ♦ User interface security
- ♦ The human element

Ensuring a secure device network is extremely important in today's world. While it is impossible to make a system completely impenetrable without making it unusable, there are many ways to improve a system's resistance to attacks.

CAUTION: Protect against unauthorized access by restricting physical access to the computers and devices that manage your building model. Set up user authentication with strong passwords, and secure components by controlling permissions. Failure to observe these recommended precautions could expose your network systems to unauthorized access and tampering.

- Software security begins with the latest software version. Patches and software upgrades should be installed as soon as they are available.
- Physical security is crucial. All computer equipment and wiring should be secured in a restricted area. Only authorized users should have access to Supervisor and controller hardware.
- If a network is configured for remote connectivity over the Internet, the most secure stations are those that are behind a VPN gateway. A station exposed on the Internet is discoverable, and vulnerable to many types of potential attacks. If a network is configured for remote connectivity over the Internet, all stations must be behind a VPN gateway. This ensures that systems are not directly exposed to the Internet.
- To put a demonstration station on the Internet, create a separate demonstration zone. Stations exposed to the Internet should not also be used to manage a device network.
- To provide network-based defense-in-depth, networks should be segmented into zones.
- All data transmission over wired and wireless connections should be secured with CA-signed digital certificates.
- If your company acts as its own CA (Certificate Authority), the company's signed CA root certificate must be separately installed in each station's **User Trust Store** and in each browser's trust store.
- The medium (usually a USB flash drive) used to store exported CA certificates and keys must be physically protected and stored in a secure vault.
- High-traffic stations (especially stations that provide public access to a controller network) must use secure **`Niagarad`** with a separate certificate from that used for the **`FoxService`** and **`WebService`**.
- Each station must be backed up regularly. Embedded systems, such as JACE controllers write audit information to a rolling buffer. To avoid losing a station's audit trail, audit histories should be regularly exported to a Supervisor station.
- Do not rely on an NTP (Network Time Protocol) server that you do not directly control. If your system's network depends on an external NTP server for the time of day, and that server is compromised or spoofed, your system may be harmed.

Security

IT managers are familiar with security issues in the context of local area networks and access management. This chapter explains Niagara's security mechanisms, which, if fully implemented and faithfully observed can prevent unauthorized access and thwart most malicious cyber attacks against a building's control systems.

Industrial control systems, of which the Niagara Framework is a classic example, have traditionally avoided security breaches by employing obscure protocols and running on in-house intranets that were not exposed to the Internet. This "security by obscurity" is no longer viable. Increasingly, data sharing, data acquisition, and peer-to-peer data exchange are standard business requirements. As data management and collection moves from large equipment into every-day appliances, preventing malicious data theft, denial of service, and command and control take-over becomes an imperative.

Data integrity and communication security are high priorities for the framework designers.

Threats and risk assessment

A building's control systems are compromised when its operations, personnel, and/or technology present weaknesses or vulnerabilities that malicious threat actors can take advantage of through intent, capability or simple opportunity. Managing such risks is a company-wide commitment.

Activities by such intruders can:

- Interrupt operations, forcing systems to stop
- Capture and modify data, including employee information, control data, and alarm data
- Store inaccurate data in station databases and histories
- Prevent systems from issuing alarms
- Interfere with communication between remote devices and the dashboard used to monitor them potentially causing life-threatening harm

Defending against these threats requires technology, best practices and rigorous standards. "Organizations must constantly adjust and refine security countermeasures to ensure protection against known and emerging threats" (From: "Recommended Practice: Improving Industrial Control System Cybersecurity with Defense-in-Depth Strategies," which is available at the Homeland Security-sponsored web site: https://ics-cert.us-cert.gov). This paper recommends "layers of monitoring and protection based on the business's exposure to cybersecurity risks."

The precautions your company needs to take against such threats depend on management's ability and willingness to implement security measures. The process begins with recognizing the threats, understanding the technology provided by the Niagara Framework to mitigate the threats, and articulating a reasonable definition of acceptable risk. This document summarizes the security features of the Niagara Framework. The white paper, mentioned above, recommends a three-pronged approach to risk assessment:

- The organization sets the company's risk management strategy: establishing risk-analysis methodologies; identifying mitigation measures; articulating the level of risk (risk tolerance) to accept; establishing ongoing risk management procedures; and specifying who will oversee the company's risk management strategy.
- Based on the company's strategy, business processes need to be put in place: defining core functions; prioritizing functions; identifying needed information, interdependencies and information flows; developing security requirements; and identifying who should be involved with what aspects of the implementation.
- Implementation is based on the life-cycle of a threat: identifying, protecting, detecting, responding, and recovering.

The Niagara Framework provides multiple levels of security for industrial control system operations. Even if your company has not yet developed a top-down security strategy, you should take advantage of all framework security features, retrofitting procedures in legacy systems and implementing robust security mechanisms in new systems.

Framework security features

The framework ensures security in several ways to protect its systems from inadvertent or malicious tampering:

- All modules are code-signed.
- The underlying station file structure is designed to support secure operations. Core software functions are stored separately from user data. Application access to data is strictly controlled.
- Access to each platform, station, and other system features require user name and password credentials.
- After a period of inactivity, stations log out automatically.
- In addition to access credentials, other features require user credentials.
- Roles assign permissions to the user based on the user category. User access to system components is limited as per the assigned role.
- Client/server authentication is provided by the industry-standard TLS (Transport Layer Security) protocol.
- Use of the TLS protocol provides data encryption during transmission.
- Data traveling from one location to another is encrypted.
- As an additional security precaution, using a web browser to view QNX platform diagnostics (known as Daemon Debug) has been disabled.

Code-signed modules

To ensure secure installation, each software module (file) is digitally signed. At run time, the system validates each module's signature. This ensures that malware cannot modify the code (core framework content) during commissioning.

Each software module is distributed with a runtime profile, designated by a suffix on the module's file name. Many modules have multiple runtime profiles. For example, the `alarm` module is distributed as three separate .jar files: `alarm-rt`, `alarm-se`, and `alarm-wb`. The suffixes `-rt`, `-se`, and `-wb` identify the associated profile. The module signature is based on the module's runtime profile and runtime profiles define what type of software module to install.

Although these features are seamless to the user, knowing about them provides assurance of system security.

Data access control

The framework's data structure is designed for secure management of core application and user data. During installation and platform commissioning , the framework differentiates between two types of files based upon the content of the files: configuration and runtime data.

Files and folders that contain configuration data reside in separate locations from files and folders that contain runtime data.

- The **System Home** directory contains runtime files, such as core software modules, the JRE, and binary executables.
- The **User Home** directory contains each station and its configuration data, including system properties, templates, option files, registries, logs and other data.
- The **Daemon user home** directory contains platform configuration data for the daemon server process.

This directory structure enhances security by denying general access to the runtime files and allowing each user access to only their personal configuration files. Because configuration data are not combined with runtime data, users do not require full access permissions for an installation. This provides the flexibility administrators need, to regulate user access.

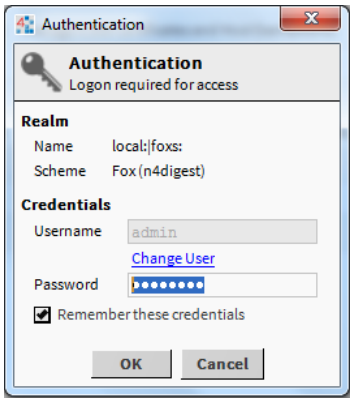
Multiple Credentials

User authentication, data encryption, and secure communication are primary aspects of framework security.

User authentication

A Niagara system typically uses login credentials (user names and passwords) to authenticate a user. Whenever a connection to the platform or a station is attempted via Workbench, first a secure communication session is established using TLS (Transport Layer Security). After a secure platform or station connection is made, the **Authentication** window opens and asks for user credentials, which are verified against the user account.

Figure 4 Station authentication window



The **Authentication** window includes a checkbox to **`Remember these credentials`**. When checked, Workbench stores the credentials in encrypted format and uses them to automatically fill in the login credentials the next time the user logs in.

NOTE: This is a convenience mechanism. However, when the box is checked, anyone with access to that Workbench is able to log in using those credentials. For highly sensitive systems and privileged accounts, make sure that the box is unchecked.

System passphrase

System passphrase protects sensitive information stored in the system's memory, hard disk and on the SD card in JACE-8000 controllers. When performing a new installation, the installation wizard prompts you to set a system passphrase for encryption purposes.

The system passphrase defaults to the factory default platform password. During commissioning, installers will be prompted to change the default system passphrase.

NOTE: Once the installation program sets the system passphrase, this step will not be presented again upon subsequent Niagara 4 installations.

To change the system passphrase, use the Platform Administration tool.

Restoring a station from a backup and transferring station files to a new location require the person performing the action to supply this encryption passphrase.

Once changed, the passphrase must be stored in a safe location from where it can be retrieved as needed.

Certificates private key

The framework supports TLS (Transport Layer Security) protocol, to provide secure networks using PKI (Public Key Infrastructure). It uses certificates to verify the identity of a server so that communication is trusted. Using digital certificates allows a seamless use of TLS (HTTPS) in a browser and Workbench, and provides both encryption (during data transmission) as well as server authentication.

A root CA (Certificate Authority) certificate is a self-signed CA certificate whose private key is used to sign intermediate and server certificates in a trusted certificate tree. If your company serves as its own CA, the

system requires the creation of a password for the certificate's private key. This should be a strong password, which should be physically stored in a secure vault.

The **Certificate Manager** prompts for the private key password:

- To export a certificate when creating a public root CA certificate
- To backup certificates with their private keys
- To create a CA certificate (root or intermediate) for importing into a client **User Trust Store**
- To create a CSR (Certificate Signing Request) for a CA certificate (root or intermediate)
- To sign a CA certificate

Password management

The framework uses passwords to authenticate station and platform users, encrypt stored data, and protect data in transmission. It is particularly important to handle passwords correctly. If an attacker acquires a user's password, they can gain access to the system and have the same permissions as the user has. In the worst case, an attacker might gain access to a super user account or platform account and compromise the entire system.

The framework provides the following features to help in securing the passwords in your system:

Feature	Description
Password strength	Ensures that users are choosing good, strong passwords. To meet the security needs of a particular system, the **Password Strength** property in the authentication scheme's **Global Password Configuration** property sheet allows for the customizing of password strength for a particular scheme.
Account lockout	Prevents a user from logging in after a specified number of failed login attempts. The **`UserService`** provides a way to customize lockout properties.
Password expiration	Prevents users from using passwords indefinitely. Password expiration settings are configured using the authentication scheme's **Global Password Configuration** property as well as by changing individual user properties.
Unique login requirement	When the Allow Concurrent Sessions property on a user record is changed from true (its default) to false, only one person may login with each set of credentials. This feature increases security by stopping credential sharing. People are forced to use unique accounts.
Password History	Prevents a user from choosing a previously-used password. Authentication schemes can be configured to remember users' previously used passwords.
Password reset	Ensures that a new user creates (resets the default) a brand new password known only to that user. The password reset feature is also useful to ensure that a new password policy is enforced for all users. The **Force Reset At Next Login** property in the user's property sheet requires a user to change their password.

Stronger passwords

Even passwords stronger than those configured by the **Password Strength** property are encouraged.

System users should be encouraged to:

- Use a mix of UPPER and lower case (cAsE SensItiVe) letters.
- Not use any part of the user account name in a password. For example, if the user account name is `ScottF`, then `ScottF!` or `ScottF123` are not good ideas (even though the last is considered to be a strong password.

- Not include the user's birth year in a password, for example `James1971`.
- Not include the word `password` in a password. For example, `Password1234` is, technically, a strong password, but it is unsafe.
- Avoid use of dictionary words, as they are commonly tried by brute force hacking applications.
- Use characters that require typing with both hands. This helps protect against somebody watching you type your password on a keyboard.
- Consider a string of words or a nonsensical phrase that you can easily remember, yet would be difficult to guess. For example: `Correct Horse Battery Staple #11`

Remember, a good password must be easy for a user to remember, yet difficult for an attacker to guess.

User interface security

Several features that are available through the system interface also provide security.

System components are protected objects. Each is grouped by category. Once a human or other station (machine user) is authenticated, authorization to access station components depends on the user's assigned role.

Each role defines a permissions map to the component category groupings. Permissions define access rights (the right to read-only, read and write, and invoke action) to each category. In addition, each role identifies which nodes of the station hierarchy are visible. The **Admin** role provides a user super-user permissions and access to all hierarchies.

The systems integrator (initial system installer) usually sets up component categories, roles, hierarchies, and users. The facility manager maintains these security constructs.

These features ensure that all users, human and machine, can access the services and components intended for them. All other services and components remain protected.

In addition to restricting access based on need-to-know (role, hierarchy and user), when a valid user remains logged in, but inactive for a period of time, the system automatically logs the valid user out to prevent unauthorized access by someone else.

The human element

Any system, no matter how well designed and implemented, ultimately relies on people. Large and complex systems are susceptible to mistakes made by inexperienced or untrained personnel, as well as the activities of malicious insider and external threats.

Even after implementing appropriate technical safeguards in the framework, system owners and users need to ensure security by adopting these measures:

- Policies that are clear and actionable set healthy expectations and lay the foundation for detailed procedures. Rules of behavior need to be clearly understood and enforced with appropriate controls and sanctions for non-compliance.
- Procedures must define secure processes and system configuration tasks that follow standards, are repeatable, and lend themselves to training new employees quickly.

  Procedures must cover the security features built into the Niagara framework, including the importance of using strong passwords and changing them frequently; securing communications using CA-signed certificates (avoiding the use of the default self-signed certificates that appear when a trust store cannot authenticate a server and cannot authenticate client-server relationships); assigning categories to devices; and limiting access based on clearly-defined user roles.
- Security-specific training, awareness of threats, and knowledge of available protection measures must be included in company-wide training programs. Industrial control system operators need to be aware of the signs of intrusion, what they should do immediately to halt the damage, and how to ensure the success of the investigation that will follow.

Management needs to be aware of the costs and benefits of the recommended protection measures so they can make informed decisions.

Chapter 3 Networking and performance optimization

Topics covered in this chapter

- About the NiagaraNetwork
- About other networks
- Networking technologies
- About connecting devices to a company LAN or WAN
- Single-site network application
- Multi-site network application
- System performance optimization

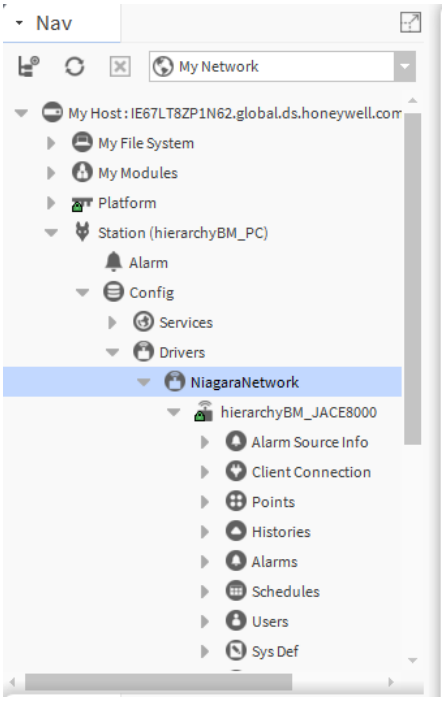
The framework's software suite implements a highly efficient adaptation of the Java component software model and current Internet technologies to provide true interoperability across a wide range of automation products. Using Workbench, this object model integrates a wide range of physical devices, controllers, and primitive control applications including LonMark profiles, BACnet objects, and legacy control points.

The device driver networks reside under the **Drivers** container in the station database and in Workbench they are installed to a station from the palette side bar.

About the NiagaraNetwork

By default, every station has a NiagaraNetwork driver under its **Drivers** container. The NiagaraNetwork in the Supervisor station models data that originates from the other remote stations in the network.

Figure 5 The NiagaraNetwork in the Workbench Nav tree



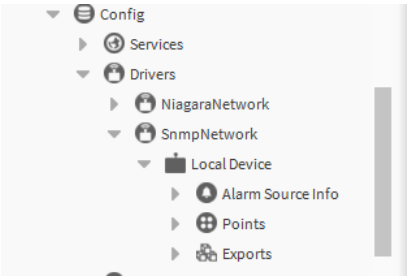
While theNiagaraNetwork uses the same driver architecture as other drivers, a NiagaraNetwork differs from them in that:

- All proxy points (BooleanPoint, EnumPoint, NumericPoint, or String Point) under a station are read-only.
- Connections between stations occur as client and server sessions using the Fox/Foxs 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 the station is server. Client authentication (performed by the server) is required in all Fox/Foxs connections.

About other networks

In a station, one or more network drivers retrieve and model real-time data values as proxy points, and lower-tier components in the driver's architecture. To support proxy points and other modeled data of a driver, that driver's network architecture must be added to the station. In addition, (depending on the driver/devices) other data items native to devices may also be integrated in the station, such as schedules, alarms, and data logs (histories).

Figure 6 A non-Niagara network (SnmpNetwork, in this example) in the Drivers container



A driver network is a top-level component in a station. For drivers that use field bus communications, such as Lonworks, BACnet, Snmp and Modbus (among many others), this component corresponds directly to a physical network of devices. Except for BACnet, the network component matches one-to-one with a specific comm port on the host platform. This port could be a serial port (RS-232 or RS-485), Lonworks FTT-10 port, or an Ethernet port.

A BACnet network component is unique because it supports multiple logical BACnet networks, which may use different comm ports. For example, if a host uses BACnet MS/TP, it may have one or more RS-485 ports.

Other non-field-bus drivers also correspond directly to the physical network of devices. For example, the Ndio (Niagara Direct Input/Output) and the Nrio (Niagara Remote Input/Output) drivers correspond to physical I/O points on a host controller or hardware I/O module (either directly attached or remotely connected).

Networking technologies

A host is either a PC running a local Supervisor station or a remote controller running a remote station. The framework supports a PC running a Win32 or Win64–based operating system and a remote controller running the QNX operating system. Both operating systems provide industry-standard networking technologies.

The framework runs on these hosts:

- A PC or laptop running Microsoft Windows, Linux, or Solaris using the HotSpot JVM.
- An embedded JACE controller using the QNX operating system and the Oracle Hotspot Java Virtual Machine (JVM).

Networking technology	Use
Ethernet	Required to configure hosts (Supervisor PC and remote controller).
TCP/IP (IPv4/IPv6): QNX-based and Windows-based hosts	Required to configure hosts. TCP/IP use includes: • IP Address (x.x.x.x) • Subnet Mask • Default Gateway.
DHCP: QNX-based and Windows-based hosts	Supported, however, most hosts require a static IP address to facilitate data communication.
DNS: DNS and Windows-based hosts	Supported in Windows-based and DNS-based hosts.
WINS: DNS and Windows-based hosts	Supported in Windows-based and DNS-based hosts.
DDNS: QNX-based and Windows-based hosts	The framework does not support DDNS in a JACE. You must use an external router or other device which supports DDNS.
Proxy Servers: QNX-based and Windows-based hosts	Communication via proxy servers is supported on all host operating systems.
Firewalls: QNX-based and Windows-based hosts	Hosts operate well in many firewall environments, provided that they are configured properly.

About connecting devices to a company LAN or WAN

There are many different ways to design and configure networks.

Each individual networking environment presents a unique challenge for maximizing network efficiency. In addition to understanding general networking principles, it is important to understand how the framework handles networking.

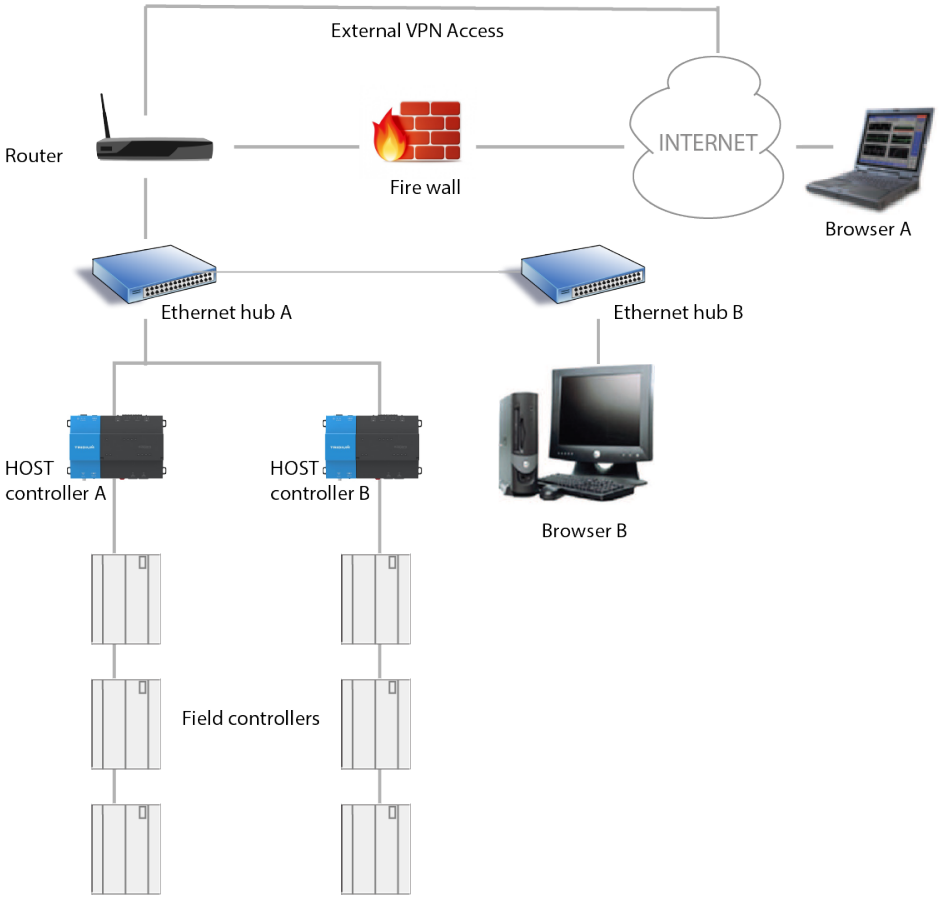
Consider this information about connecting framework devices to a LAN or WAN:

- Proxy points are designed to be used across connections that are always available (so that connections between linked hosts fail infrequently). Therefore, proxy points should only be used on a LAN or WAN, and only on a LAN/WAN that provides a reliable connection between hosts.
- Access to hosts with private IP addresses can be made from hosts on the same LAN/WAN. External hosts can only access privately-addressed internal hosts through a virtual private network (VPN).
- The key to success in many installations is early involvement by the IT department at the site. If devices are to be exposed to the company's LAN/WAN, all IT policies must be followed.

Single-site network application

In this example, a customer has a single site with a LAN, exposed to the Internet through a firewall.

Figure 7 Typical single-site (LAN) architecture



The network is configured for remote connectivity over Internet. All the stations are behind an Access point TLS VPN gateway, which ensures that systems are not directly exposed to the Internet.

Browser A is a BUI (Browser User Interface) user located across the Internet and browser B is a BUI user located internally on the LAN. The site has multiple field controllers controlling field devices. These field controllers are remotely controlled by controller A and controller B. Each field controller has a private IP addresses, so they are not accessible by the Browser A located across the Internet. However they are available to Browser B located on the same LAN.

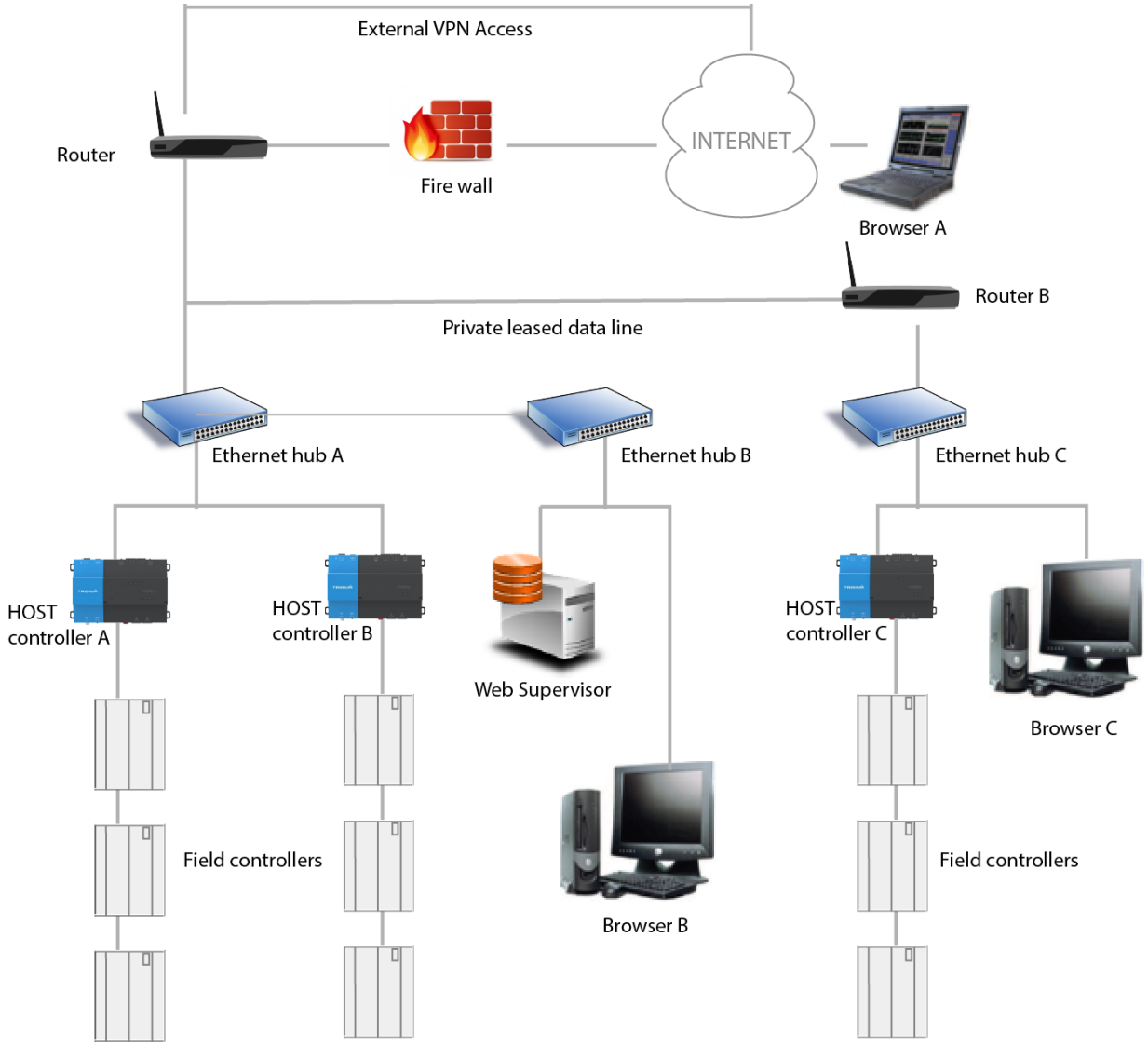
The Supervisor station is connected through Ethernet hub A and has a public IP address assigned to it in the firewall. It can be reached by both Browsers A (the external user) and Browser B (the internal user). In the example, the Supervisor has been engineered to include graphics that show real-time information originating from controller A and controller B. To accomplish this, the Supervisor proxies data in the controller A and controller B. In addition, the Supervisor functions as a supervisory station, archiving the other station's (controller A and B) data logs, alarms, and so on. This data is available to both Browser A and B.

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

Multi-site network application

The below multi-site network example shows the major characteristics of a typical Niagara system architecture.

Figure 8 Typical multi-site (WAN) architecture



In the example, the company added a remote host to a LAN at another site (site 2). The sites 1 and 2 connect to each other using a private data line that is leased from a phone company, thereby creating an enterprise-wide WAN. The Access point VPN gateway ensures that the systems are not directly exposed to the internet. The network is configured for remote connectivity over internet.

The framework proxy points update the live graphic presentations on the Supervisor. The network site administrator decided to move the Supervisor onto the enterprise LAN at the site 1, since there is more data traffic generated internally than the occasional external browser (Browser A) use. The internal IP address of the Supervisor changed to one in the IP range of the new network to which it is attached, but the external address (because of the firewall) did not change.

System performance optimization

The goal of performance optimization is to configure system capabilities and integrate subsystem elements so that all components operate at or above user expectations.

What impacts the performance of the system?

Many factors impact the performance, including: the number of devices on the network, the frequency of accessing points, the size and frequency of generated reports, the complexity of analytical algorithms, the scheduling of station backups, the security overhead, and the frequency and complexity of BQL (Browsing Query Language) queries.

How can we improve system performance?

System optimization is a process of balancing memory resources: CPU memory, which is the main resource; VM (Virtual Machine) memory or Heap storage, which loads and runs the system at boot; and System memory, which runs the operating system. Each memory resource must have a reasonable amount of reserve for the system to function optimally. Performance degrades dramatically when all memory resources are at maximum utilization.

At times, as the CPU performs some operations it's memory usage spikes to 100 percent and after completion it settles back down. Running the CPU all the time at 100 percent may cause it to crash.

NOTE: As a safe practice, it is recommended to generally keep an embedded platform's memory usage within the below given limits.

- CPU Usage < 75% average
- Heap Usage < 75 % of available
- Free System Memory > 7 MB

There are many factors that contribute to the usage of the above three key indicators that are difficult to quantify for a given set of drivers, histories, alarms, reports, database configuration, and User Interface.

A station running on a Java VM creates objects and marks them for cleanup. Java runs garbage collection and cleans out all the objects that were marked in the VM memory so the whole process can start again. You can see the VM memory increases and decreases. You must have that cushion of free memory so that Java VM can operate effectively.

Performance optimization is a moving target. At times your system may have plenty of VM memory available, but you cannot increase the station size due to the fact that the controller is running low on CPU (too many changes happening in the station). Or your CPU may almost be idle, but you may be running out of VM memory. The only solution is to configure the station such that all three resources are within the recommended values.

Since every system is programmed differently, a specific device count cannot be determined. One system runs a JACE-8000 with 265 BACnet MS/TP devices, however, another system that puts graphics on a JACE-8000 may have 100 BACnet devices. Or changing how many points are brought into the station changes the equation for device count. So the performance for both the systems may vary. By managing the usage of CPU, VM memory, and system memory each system can optimize its performance.

Index

V

W