TRIDIUM™
revolutionary://software.solutions™

The Niagara Framework

Control System Interoperability with Seamless Intranet/Internet Enterprise Connectivity

Tridium, Inc.
3951 Westerre Parkway
Suite 350
Richmond, Virginia 23233
(804) 747-4771
http://www.tridium.com

Ernie Allen, Technical Services Manager
John Bishop, Regional Manager

The Niagara Framework

Control System Interoperability with Seamless Intranet/Internet Enterprise Connectivity

Introduction

Tridium, Inc. designs and markets Java-based automation products and services with built-in Internet connectivity to a broad range of distribution partners in the building automation, energy services, power/utility, and industrial sectors. Tridium's Niagara Framework brings together the ongoing computerization of control applications under the umbrella of a single, integrated system architecture. The suite of Niagara component software applications supports true plug-and-play, multi-vendor interoperability, resulting in significantly lower automation and information infrastructure costs and was designed from day one to take advantage of the power of the Internet.

From an engineering perspective, the Framework is built on an enhanced JavaBean object model that solves real world problems associated with a distributed real-time control system. As such, the Niagara architecture provides pre-built, pre-tested, tried-and-true building blocks that can be used to create and modify control solutions quickly. These components are extensible. They can be used as-is for meeting specific automation requirements or new components can be easily created using Niagara's integrated application development environment.

From a solutions perspective, the main reason that automation and information infrastructures become obsolete is not that they wear out, they simply fail to adapt to the changing environment around them and do not integrate with new applications and evolving technologies. The Niagara architecture provides an open component framework, which makes it possible to build next-generation, multi-vendor, interoperable control systems that satisfy the need for Internet openness, while maintaining the determinism and integrity required for real-time control. As a result, the Framework provides a supportive environment in which the infrastructure is allowed to adapt, where multi-vendor systems integrate easily, and one that reduces overlap, rework, and premature obsolescence.

Contents

With Sidebars On:

From an end-user perspective, the Niagara architecture puts the user in control of their facilities, letting them choose the products they want, with the features they need, from the vendors who perform to their expectations. The Framework allows end-users to quickly reconfigure their control systems dynamically over the Internet in response to changing needs. For the first time, end-users do not have to settle for a compromise solution selected from traditional, one-size-fits-all automation products that never fully solve unique control problems.

Primary Design Considerations. Two primary design accomplishments allow for such an innovative solution. First, each Niagara software component is a self-contained unit with specific functionality and a well-defined interface to the Framework. This means that applications built from components can be trusted and they allow unlimited, simple extensions and modifications. Moreover, due to their well-defined interface, components can communicate with one another, readily pass information, and changes can be made while the control application is running. The component framework can be used to express a variety of constructs that represent physical devices, controllers, and primitive control applications. These include, but are not limited to LonMark profiles, BACnet objects, and legacy control points. This protects the end-user's investment by allowing legacy systems to be brought forward, where they can readily adopt new standards, solutions, and applications.

The second design innovation is that of embedding open Internet protocols at the controller level. This feature allows the Niagara architecture to provide a single point of connection to both the Internet and the real-time control system. By doing so, the Framework allows access for multiple sites from virtually countless users at a cost structure that is remarkably low. The end-user finally has a solution that makes it possible for anyone with appropriate security, from anywhere, at any time to interact with smart devices embedded in building automation, power/utility, and industrial control networks through a standard Internet browser.

Tridium's key business objective is to leverage the strengths of many alliance partners to promote vendor-independent, robust control solutions that integrate seamlessly with a wide variety of business and information applications in a market where traditional means of distribution and support are changing. The goal is to provide innovative solutions for the way control systems are built and distributed by empowering customers and putting them in charge of their product and service choices. By striking strategic alliances with *open*-minded partners, interoperability and ubiquity are ensured. By building the design on emerging protocol standards, then embedding that solution at the controller level, Internet openness becomes an inherent part of Tridium's distribution, deployment, and support strategies.

Customer Benefits. The benefits of adopting the Niagara Framework as a complete and comprehensive solution include: 1) it provides end-users the openness they demand in controlling their facilities; 2) it provides System Integrators a complete set of tools and services to assist them in the integration process; 3) it allows true integration and support of both the control environment and the information domain; and, 4) it is a unique and effective solution that is appropriate in virtually any segment of the automation marketplace.

Overview. This paper is a non-technical overview of the Niagara Framework. It is written for businesspersons that may be evaluating the Framework as their integration solution. It is written as a general introduction to the enabling technologies on which Niagara is founded. The sidebars and glossary provide background information to support the general discussion and to give the reader a more thorough understanding of related topics.

Trends in the Control Systems Market

Technology enablers are creating a convergence toward a common system architecture for different types of control.

Recent general acceptance of building automation and device communications standards such as BACnet and LonWorks permit, for the first time, seamless interoperability between new building automation services from a variety of suppliers. Concurrently, the recent emergence of enterprise component software standards such as CORBA (Common Object Request Broker Architecture), DCOM (Distributed Component Object Model), and XML (eXtensible Markup Language) have created an infrastructure that allows information to be shared between building automation systems (BAS) and enterprise information systems (EIS). Finally, development of Internet and Java computing standards continue to permit rapid development of portable control and information management applications for multi-platform Internet environments.

Building automation systems automate the many services required to successfully operate and manage facilities including environmental control, fire and life safety, lighting, energy management, and security access control. These services have traditionally been provided as independent, standalone systems by multiple, proprietary vendors. It is very common for a single hospital facility to have 2-5 independent automation systems, while a national chain might have ten or more systems throughout the country. While improvements in systems integration have occurred over the past ten years, only recently has it become cost effective and, in some cases, technically possible for a user to deploy an easily portable and fully integrated facility management solution.

Enterprise information management systems have rarely been linked to the BAS due to differences in communications standards, divisions in reporting responsibilities, or concerns surrounding data integrity and system security. However, much of the BAS data eventually ends up in the enterprise database through a manual process. With the advent of power deregulation and the widening of services provided by facilities management companies, large users of the EIS are expressing the desire to automate the retrieval of data from the BAS and have the two systems integrate more seamlessly.

Internet-enabled user interfaces have become quite popular in recent years, but heavy investment by traditional BAS companies in proprietary software has slowed the availability of cost effective, multi-user, and multi-platform solutions. Most of the existing BAS systems were not originally designed to take advantage of the Internet's portability. These systems require that a full set of BAS software reside on each computer that has access to the system. However, with Java software embedded at the controller level, a limitless number of users can access the control system over the

The Open Systems Movement

One of the first applications of device communications was that of a thermostat relating a setpoint and process feedback by means of mechanical action. Pneumatic devices of the 50's and 60's used a 3 to 15 pound per square inch (PSI) signal as the standard protocol to communicate setpoint and process feedback to heating and cooling equipment often manufactured by different vendors.

When electric devices took over the market in the 70's, the 4-20 mA signal became the open protocol standard. As microprocessor-based devices became popular in the 80's, each manufacturer, without regard to other manufacturers, developed devices that used digital signals to communicate.

In the 90's, customers demanded that devices produced by different manufacturers actually work together. As a result, manufacturers began to publish their communications protocols and independent associations evolved to refine and manage these published standards.

Today there are several industrial protocols that are published and managed by independent associations. These include DeviceNet, which is supported by more than 270 manufacturers as

Internet at no additional cost per seat.

Traditional channels of distribution are changing in the commercial controls marketplace.

As building automation and device communications standards have created a momentum in the industry to reduce dependence on proprietary building automation systems, independent System Integrators are replacing the traditional BAS manufacturer branch structure as the dominant distribution channel. With access to multiple vendors' products and a reluctance to commit to a single vendor, System Integrators require an independent set of tools and services to assist them in the integration process.

Finally, the deregulation of the utility industry is contributing to the unheralded change in the distribution channels of the commercial controls market. As power and gas companies look for new growth opportunities, many are choosing to become Energy Services Providers, a service once reserved for traditional BAS manufacturers. With the varied needs of typical utility customers, combined with a wide range of legacy systems, utility companies are looking to provide vendor-neutral solutions.

The Problem with Integration

Building owners today are faced with an ever-increasing number of intelligent systems that are being installed in their buildings. These systems are not limited to HVAC systems. They include lighting systems, alarm and security devices, medical safety equipment, power metering, and more. It is no longer acceptable to rely on a single manufacturer for field bus devices, connected by an area controller, and accessed through proprietary operator interface software. Furthermore, building owners are not willing to replace existing systems in order to adopt new standards. Enhancements to automation systems must easily integrate legacy products and move their functionality forward.

Today's microprocessor-based systems offer tremendous benefits in terms of precise control and data gathering capability, but these benefits present challenges too. The greatest challenge presented by such a selection of affordable and impressive devices is that of integration. Not only do traditional proprietary controls fall short of offering customers the openness they demand in controlling their facilities, but even the network management services and tools that have recently promised open integration of multi-vendor devices reek of propriety.

System Integrators are positioning themselves to support multi-vendor products in an effort to provide best-of-breed solutions for their customers. Because of their reluctance to commit to a single vendor, System Integrators are turning to vendors that provide systems designed around standard protocols. But, the problem with most BACnet and LonWorks solutions is that although they deliver on the promise of open systems integration, they are

members of the Open DeviceNet Vendor Association (ODVA). In addition, the Fieldbus Foundation has taken the lead in managing the development of a single international, interoperable fieldbus standard. The Fieldbus Foundation was established in September 1994 by a merger of WorldFIP North America and the Interoperable Systems Project (ISP) and it consists of nearly 120 of the world's leading suppliers and users of process control and manufacturing automation products.

In the commercial controls market, there are two protocols that are considered to be the leading open standards. These are BACnet and LonWorks. The LonMark Interoperability Association was founded in August 1994 by 36 companies and is an independent organization of over 200 members including virtually every major controls company in the BAS industry worldwide.

Developed under the auspices of the American Society of Heating Refrigeration and Air Conditioning Engineers (ASHRAE), BACnet is a data communications protocol for BAS networks and is now an American national standard and a European pre-standard, with the potential of becoming a global standard.

designed around proprietary network management tools.

Integration tools offered in the building controls market today do not provide the complete and comprehensive solution that System Integrators are looking for to support their customers. A truly comprehensive solution must combine common network management services for both open standards devices (BACnet and LonWorks) and legacy products with a full-featured environment that blends the control system seamlessly with the enterprise information system.

In addition to network management support for both open standards devices and legacy products, a comprehensive solution must include:

- Full graphical user interface.
- Synchronization of controller databases and database storage and backup.
- Password protected access.
- Data collection and enterprise-level information exchange.
- Synchronization of global time functions and central scheduling.
- Alarm processing and routing.
- Extensive support for all standard energy management functions.
- Customization for non-standard control applications.

The Niagara Framework solves the many problems associated with interoperable systems being implemented with today's device communications standards. By embedding the network management function at the controller level and combining that with Internet connectivity, the Niagara Framework provides a solution that allows not only access, but also modification of control system parameters from any Internet connection. This is combined with a feature-rich desktop software suite that includes an extensive library of applications and protocol drivers, network management tools, database administration functionality, and a graphical real-time control system user interface that finally delivers to building owners the cost-effective and functional solution to the problem of integration.

Control System Integration Today

Open protocol and *standard protocol* are terms often used to discuss two very different methods of achieving interoperability. Open protocol means that the communications language, which is typically proprietary, used to exchange information between microprocessor-based devices is published and openly available. This makes it possible to develop an interface or gateway to products supporting that protocol. The difficulty with gateways, though, is that they represent bottlenecks that can adversely affect the ability of systems to share data.

Standard protocols, on the other hand, refer to communications methodology that is developed through a public and cooperative process, resulting in a non-proprietary language that all manufacturers have agreed to equally. Standard protocols allow products developed by different manufacturers to communicate without the need for a gateway, which can lead to true interoperability.

The Niagara Framework Solution

The Niagara software suite implements a highly efficient adaptation of the JavaBean component software model and Internet technologies to provide customers with true interoperability across a wide range of automation products. As a subset of the complete Framework, 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.

Business Today

Today, most companies have overly complex automation and information technology infrastructures that are expensive to manage and maintain. These systems are typically closed and proprietary, which makes them difficult to operate and virtually impossible to enhance. The complexity and inefficiency of these systems accounts for the high cost of infrastructure support – identified to be as much as 80 percent of the total cost of the automation and information technology.

Historically, the building automation, energy services, industrial automation, and utility industries have been distinct businesses supported by proprietary, standalone products that have made integration very difficult. But, technology is changing the fundamental nature of these standalone products, making them parts of larger systems with common open system architecture. Open industry standards are imposing order on these multi-vendor systems and making interoperability possible. The control system market is on the cusp of a major technology cycle as vendors move to adopt the openness of these enabling technologies.

The older monolithic systems supplied by single vendors are being replaced by automation and information solutions developed to open industry standards, by strategic alliances, and through partnerships and collaborations. As a result of this strategic partnering, Tridium is leveraging the critical capabilities of each partner, increasing the flow of innovation and improving responsiveness to market changes and shifts in technology. These changes are being driven by the need for rapid application development, enterprise integration, and secure network-centric applications – capabilities nearly impossible to provide in a world of proprietary products and protocols.

The Nature of the Solution

The primary goals of the Niagara Framework are to provide solutions to several key issues that continue to plague the control

Device Communications Standards

Device communications standards such as BACnet and LonWorks are becoming widely accepted in the BAS community and are, for the first time, permitting seamless interoperability between new BAS services from multiple vendors.

BACnet. Building Automation and Control Networks

BACnet is the ASHRAE/ANSI Standard 135-1995 that was adopted in August 1995 as the data communications protocol for computer equipment used for monitoring and control of HVACR equipment and other building systems. This protocol models each building automation and control computer as a collection of data structures called objects, the properties of which represent various aspects of the hardware, software, and operation of the device. These objects provide a means of identifying and accessing information without requiring knowledge of the details of the device's internal design or configuration.

LON. Local Operating Network

LON is an acronym coined by Echelon Corporation and refers to an intelligent control network that facilitates

systems industry. These include:

- Successful integration of multi-vendor building automation products.
- Seamless integration of BAS data with enterprise information systems.
- Ready and secure access to multi-site, multi-user, and multi-platform control systems that take advantage of cost effective, Internet-based user interfaces anywhere in a global organization.
- A comprehensive, object-oriented solution that supports rapid application development. One that integrates the entire software development process from planning and analysis, through design, construction, and maintenance.

Tridium's approach to automation technology embraces many industry standards to achieve openness and integration. However, open standards alone are not sufficient for a distributed control system environment. There is also a critical need to protect the integrity and determinism required for real-time control.

System performance in the control domain must be deterministic. In other words, functions must occur exactly the same way and at exactly the same speed every time they are executed. For example, the end-user must have guaranteed delivery of an alarm within a specific time. Systems operating in the control domain must be physically robust, with the ability to support redundant components. They must also be error-tolerant to protect against operator error. For instance, if the operator enters a setpoint that is out of the allowed range, it is refused.

Tridium has developed a unique approach and implementation that provides a multi-vendor, interoperable control system that satisfies the need for Internet openness, while maintaining the determinism and integrity required for real-time control. The Niagara Framework meets these demands by partitioning the open Internet environment and the deterministic real-time control environment, providing single point access to both the Internet and the real-time control system from any Java-enabled browser.

The Niagara Framework is based solely on the Java computing model to insure deterministic control interoperability of software components and portability to all control and server platforms. It also includes special real-time drivers to integrate intelligent embedded devices over multiple field bus communications protocols into a common development and deployment environment that enables interactive and incremental customization in a manner that suits the skill level of the user.

Contrary to the control domain, the information domain is data-centric. It is only concerned with data and the information that can be derived from that data. It is a transaction-processing environment, where files are updated over the Internet as new

communications between a group of devices that sense, monitor, communicate, and control. LON networks are being used today in various market segments including building automation, industrial automation, power/utility automation, and transportation. Common applications include HVAC systems, lighting systems, alarm and security devices, medical safety equipment, power metering, load management, and so on.

There are a variety of LON-related terms that are used and confused within the industry today. These include:

LonMark. Refers to the mark signifying that a product has met LonMark guidelines that allow it to interoperate with other LonMark devices on a LON. LonMark certification is granted through the LonMark Interoperability Association.

LonTalk. Refers to Echelon's open-architecture communications protocol used by all LonWorks-based devices. The LonTalk protocol implements the entire seven layers of the OSI model using a mixture of hardware and firmware on a device known as a Neuron chip developed by Echelon.

LonWorks. Refers to the collective hardware and software technology developed by Echelon to provide an off-the-shelf, peer-to-peer networking technology platform for designing and implementing interoperable control networks.

data is collected and information is displayed as queries are made. These activities do not usually happen in real-time, but are more batch oriented.

The information domain emphasizes audit trails for tracking changes in data, the time they were made, and the person who made them. Security features, such as firewalls and powerful encryption and authentication techniques, are designed to safeguard the information and protect against unauthorized access.

The information domain does not require Java portability. It does, however, require control system connectivity to many non-Java legacy applications. The Niagara Framework makes extensive use of enterprise-level standards and other industry standards to communicate control system information to the enterprise legacy applications, which use non-Java object models. For example, with the Framework, the end-user can monitor manufacturing plant energy use and cost, provide direct control of the equipment, and pass energy cost data directly to the enterprise-level cost accounting system.

The dynamic, object-oriented nature of the Niagara solution allows the creation of high-value, complex applications easier and faster than traditional 'compile and load' systems. It allows the user to create control logic, test it in real-time, and make incremental modifications very quickly. With the ability to implement and test new control sequences swiftly, the user can go through several iterations producing finely tuned applications that satisfy specialized requirements in less time.

The Niagara Solution

A new set of technological enablers has the potential to simplify the automation and information architecture across the entire enterprise. New system solutions powered by this flexible enabling technology provide companies with an opportunity to significantly reduce their infrastructure costs. Combined with the benefits of architectural simplification and application interoperability, these improvements are positioning participating companies to better face escalating global competition, more demanding customers, and accelerating technological changes.

It is important to note that the same technological enablers that promise to simplify the infrastructure, reduce its cost, and make it truly interoperable are creating a convergence toward a common architecture for different types of automation and information management systems. Technological convergence is breaking down barriers between different industries and driving them together, particularity those based on electronics and information technology.

The Niagara Framework is the control systems industry's first software technology to integrate BACnet, LonWorks, and various Internet standards into a common object model application environment embedded at the controller level and supported by a

Component Software

What is a component? A component is a reusable software building block. It is a pre-built block of encapsulated application code that can be combined with other components and 'fresh' code to form an application, which dwells within a container. Containers provide an application context for the collection of components – they provide the management and control services needed by the components to execute. In real terms, containers provide access to operating system processes (threads) in which to execute the component.

A component model defines the basic architecture of a component. It defines a set of rules that are used in developing software modules that specify how the component is to interface with its container and other components. The goal of a component model is to provide for rapid development of reusable software modules in such a way that components developed by multiple vendors, with various application development tools and runtime environments, running on different computer platforms, can assemble them into applications without the need to recompile.

Components come in various sizes. Client components are typically executed within some type of visual container and are relatively small. An example of a client

standard Web browser interface. The Framework includes integrated network management tools to support System Integrators in planning, designing, configuration, installation, and maintenance of BACnet, LonWorks, Internet, and various other system networks.

The Niagara architecture is very scalable. In its minimum configuration, it can support a control system for a small building running a network of BACnet and LonWorks devices connected to a single Niagara JACE controller on the same field bus. The JACE controller is a Java Virtual Machine (JVM) that provides the environment to manage and run the control system database with field bus and enterprise LAN connectivity. Since the station database is composed of Java objects, it can easily run on multiple platforms ranging from a network computer that supports embedded systems to a desktop server platform that integrates multiple systems.

Being highly scalable and multi-platform, the Niagara solution can also be configured to supervise a network of JACE controllers connected over Ethernet and can support unlimited users remotely connected over the Internet. For corporate customers, the Niagara Framework can be configured to integrate the automation system with the enterprise information infrastructure and/or the industrial and power system architectures.

Because the Niagara Framework is based on the JavaBean component software standard and the Internet, its cost structure is remarkably low. Not only are multi-user Niagara solutions competitive with today's proprietary single-user BAS workstation packages, there is no extra cost for adding more users. With typical Windows-based solutions there is additional cost for each new operator workstation.

Niagara Components

The Niagara architecture is a suite of software built on the JavaBean object model, enhanced to solve real world problems associated with a distributed real-time control system. The core of the architecture is a flexible framework designed to integrate heterogeneous devices and protocols into a common distributed infrastructure using plug-and-play software components.

Out of the box, Niagara may be used to implement widely diverse building automation solutions. Niagara fully supports BACnet and LonWorks as well as highly flexible objects for alarming, scheduling, logging, and Web access.

In addition, the Niagara Framework may be used to create innovative solutions for other vertical markets. As a total, integrated solution for generic distributed control systems, Niagara provides a proven foundation on which to build new applications. Moreover, solutions created with the Niagara Framework are guaranteed to interoperate with Framework applications from other vertical markets.

component may be a simple graphical control like a button. Server components, on the other hand, are application components that run on a server and they can be rather large and complex. For instance, a database management component contains database requests. Multiple database clients submit their requests concurrently and rely on the container to process the transactions. From simple to complex, components provide a standard interface that enables other parts of the application to invoke its functions and access the data it contains.

Enterprise Component Software Standards

During the last few years of the twentieth century, technology on the shop floor of competitive manufacturers has advanced by leaps and bounds. New concepts in manufacturing have revolutionized the way many manufacturers operate. Computerization in the manufacturing environment and in the manufacturing process has created a revolution where on-line transaction processing (OLTP) has become a familiar phrase, promising new levels of efficiency. The need to integrate the technology of the shop floor with advanced Enterprise Resource Planning (ERP) systems is bringing productivity to levels undreamed of only a few

The Niagara Object Model

The foundation of the Niagara architecture is its object model. The Niagara object model is an enhanced version of the standard JavaBean object model. The model has been extended to deal effectively with the challenges presented by a distributed real-time control system.

Nodes. All Niagara objects derive from a common base class: Node. A node is the basic building block of the control system and it has all of the fundamental characteristics of a JavaBean – it has a set of properties it exposes, a set of methods that it allows other objects to call, and a set of events that it can fire. Examples of nodes include Analog Output, Schedule, LonWorks device, graphics object, and Control Engine Service.

Each node provides all of the core functions used to manage an object in the system. These include:

- Naming. Both a 32-bit handle and a simple yet unique system-wide identifier that resembles a standard URL (Uniform Resource Locator) identify all nodes. The 32-bit handle provides an efficient way to store link references and provides fast object lookup. Whereas, the URL-like identifier provides the primary means for lookup in an Internet-enabled system composed of multiple stations.

- Properties. Properties are named attributes that define object behavior or appearance. Properties can be written to and read from. They can be presented on a property sheet to be customized by the user, exposed for use by other objects, and accessed programmatically. Typically, properties are persistent, in that they are stored to disk or flash memory.

- Serialization. Object serialization is the process that transforms an object into a stream of bytes. It is a process that is necessary for persistence. In other words, to save instances of objects to a file or to a binary database, the system must be able to transform them into a stream of bytes and back again.

- Persistence. Objects are expected to serialize and de-serialize themselves for the purpose of being written to disk or flash memory for non-volatile backup.

- Linking. A key part of distributed control is sharing data. Niagara objects use links between input and output properties to bind the objects via the properties they expose, to provide access to their methods, and to notify one another of events.

- Security. Java-enabled solutions are intended for use in networked, distributed environments where security can be an issue. To that end, security groups and privileges are implemented to provide a level of tamper-free operation.

decades ago and is driving the standardization of enterprise component software.

Companies faced with the need for increased productivity need to integrate the real-time collection and processing of data about orders, inventory, manufacturing, and shipping to provide a degree of control over their operations to maintain a competitive edge. In an increasingly competitive world, the power to know what you need to know at the touch of a button can make the difference between success and failure.

Several organizations have been created to help foster the development of technically excellent, commercially viable, and vendor independent specifications for the standardization of enterprise information. The Object Management Group (OMG) was founded in May 1989 by eight companies and in October 1989 began independent operations as a non-profit corporation. OMG now has over 800 members. OMG is establishing CORBA as the "middleware that's everywhere" through its worldwide standard specifications including CORBA, IIOP, Object Services, Internet Facilities, and Domain Interface.

The Niagara Station

The Niagara station is a JVM that hosts the running of nodes. It provides the environment to configure, manage, and run a single database of nodes and the services required to support a control application. Since the database is composed of JavaBean-like objects, the JVM can easily run on multiple computing platforms.

PRISM. PRISM is an acronym for Persistent and Real-time Information Synchronization Manager. It is the heart of every station and it provides the following functions while managing its database of nodes.

- Lookup. This function manages object lookup by handle, within the station, and by system-wide identifier (SWID), in a multi-station configuration.
- Persistence. This function manages persistent changes to node properties by flushing the node to non-volatile storage (disk or flash memory).
- Lifecycle. This function manages operations related to adding and deleting nodes, loading and initializing nodes, creating and deleting links, invoking servlets, and so forth.
- Synchronized Proxies. This function provides a mechanism for one system to act on behalf of another (on a network) when responding to database requests. A synchronized proxy, or mirror image of the station database, is established locally (on a user interface workstation) to provide responsiveness at that workstation. The station synchronizes database changes.
- External Links. On a network, this function manages links between nodes in different stations.

HTTP Server. Network communications in the Niagara architecture is by way of HTTP (HyperText Transfer Protocol) running on top of TCP/IP (Transmission Control Protocol/Internet Protocol) rather than RMI (Remote Method Invocation). HTTP is more effective in dealing with establishing communications over the Internet where security measures (i.e., firewalls) may be in place to guard against unauthorized access to or from a private network.

JACE controllers run a dedicated HTTP server that manages all Niagara network communications. The HTTP server is a high-performance, full HTTP 1.1 embedded server with a standard Java servlet API (application program interface). HTTP servlets are typically more efficient than CGI (Common Gateway Interface) programs in most applications because of better performance, flexibility, portability, and security. All data requests and responses are made in MIME (Multipurpose Internet Mail Extensions) format, which allows the servlet to consult arbitrary sources of input data, then return data in a form appropriate to the

What Are the Enterprise-Level Standards?

There are numerous enterprise-level software standards such as CORBA, HTTP, XML, and others that allow information to be shared between the BAS and the enterprise information system (EIS).

This section provides an overview of the various enterprise-level standards that are affecting the control systems industry.

Object-Oriented Standards

Object-oriented design makes it possible to provide reusable software ICs that can be manipulated visually. The following standards provide support for software constructs known as objects. The following discussions include CORBA, DCOM, DDE, IIOP, OLE, and RMI.

CORBA. Common Object Request Broker Architecture

The Object Management Group (OMG) introduced CORBA in 1991 as a set of rules that establishes the client-server relationship between objects. The ORB is the middleware that enables a client object to invoke a method on a server object. The ORB intercepts the call and becomes responsible for

particular request. Examples include HTML (HyperText Markup Language) requests and various graphics and data formats including GIF (Graphics Interchange Format), JPG (Joint Photographic Experts Group), and MPEG (Moving Picture Experts Group).

Configuration Database. The station hosts a configuration database of persistent object information that may be supplemented by a disk-resident relational database of application data generated by nodes at runtime.

The configuration database is stored to disk or flash memory by way of a persistence scheme known as object serialization. Serialization is the process of encoding objects into a stream of bytes and the subsequent reconstruction of those objects. It is used for lightweight storage as well as communication via sockets. The persistence scheme used by the Niagara station is a superset of the externalizable interface offered by the standard JDK (Java Development Kit) interface, which is a customizable methodology for serializing objects. By refining the serialization process, Niagara realizes an order of magnitude performance improvement during all network communications and persistence operations. Configuration databases can be exported in whole or in part to either XML format, for easy exchange, or SNS (Serial Node Set) format, for highly efficient storage and retrieval.

Relational Database. Niagara nodes generate application data during station execution (runtime). Some of the runtime data becomes part of the node's property sheet and is stored to the configuration database. Other data, however, is better suited to a relational database model. This type of data is typically non-configuration data. Trend data and alarm histories fit this model.

On PC-based stations that support access to a physical disk drive (rather than flash memory), Niagara includes an embedded Cloudscape relational database management system known as JBMS. Cloudscape's JBMS is a fully functional object-relational data manager written in Java and designed specifically for embedded server-side applications. It provides standard SQL (structured query language) access to all application data generated by Niagara nodes. This provides the end-user with an environment in which enterprise information and the control system can share a common view of the data. Through open database protocols including ODBC (Open DataBase Connectivity) and JDBC (Java DataBase Connectivity) or via the station's HTTP server, both information and automation system applications can access a relational database that looks like it was designed specifically for a particular application.

Services. A service is a special type of node that provides access to published and verified sets of functionality for other nodes. Niagara includes a wealth of standard services for event handling, logging, Web access, and more. Examples of services may include control engine services, which execute control nodes on a periodic basis; database services, which provide standard SQL access to all application data; e-mail services, which manage

finding an object that can implement the request, pass it the parameters, invoke its method, and return the results. The client does not have to have knowledge of the server's location, its programming language, its operating system, or any other aspects that are not associated with its interface. The ORB provides the interoperability that is needed to support multiple object systems.

There are several implementations of CORBA, the most popular of which is IBM's SOM and DSOM architectures. Two competing models have been introduced from Microsoft (COM and DCOM) and Sun Microsystems (RMI).

DCOM. Distributed Component Object Model

DCOM grew out of the Component Object Model (COM), originated by Microsoft, and it extends that model to support objects distributed across a network. DCOM is a Windows-based model that competes with CORBA. It allows developers to create objects and have other programs and objects operate on them in a binary-standard manner. C++ objects, for instance, exist only for the programs in which they were compiled, whereas DCOM objects can be accessed by any COM-compliant application. Both OLE and ActiveX are COM-compliant.

e-mail routing of alarms; and others.

In addition, sets of network protocol layers, known as protocol stacks, are implemented as services and can be added on the fly. These stacks provide a common and secure means for transmitting data between networked devices. They determine the type of error checking that is used, the data compression method, and how a device indicates that it is ready to send or receive data. This architecture provides a very flexible and powerful vehicle for integrating new building automation equipment from a variety of suppliers with existing systems. Among these are services that support BACnet objects, LonWorks devices from virtually any control manufacturer, industry standard DDE (Dynamic Data Exchange), GE power monitoring devices, Johnson Controls Metalink DDE server, Ingersoll Modbus devices, and others.

Services reside in a table and are loaded with the station database so objects can find them quickly. During station initialization, each node registers itself with the services it needs and then simply pulls data as it is needed during execution.

Niagara Objects. The Niagara Framework includes objects that model themselves after standard object-oriented programming structures, which combine both data and procedures to create re-useable, self-contained entities. Although there continue to be objects and protocol drivers added to the standard library of supported devices, the core set of Niagara objects are typically categorized into five groups: control objects, applications, user interface objects, containers, and notification services.

Control objects include a variety of constructs that represent physical devices, controllers, and primitive control applications. Examples of control objects include Analog IO, Binary IO, Multi-State IO, Math Operators, Logic Operators, PID Loop, and more. Niagara's standard library of objects is based on the BACnet model.

A more complex set of objects, categorized as applications, is available for scheduling, logs, and custom sequencing. Program objects, which are included in this category, provide compact, easy-to-use, and very powerful software components that allow the user to customize and otherwise extend standard control solutions for specific needs.

Program objects are created using a BASIC-like programming language that is used to declare inputs and outputs, assign configuration properties, and define object behavior. Users have access to an extensive library of functions including math and logic functions, string manipulation, e-mail, and more.

User interface objects provide a set of animators and graphical components that can be used to build highly visual windows that represent control sub-systems. User interface objects include a bar graph, time plot, image spectrum, text box, hyperlink, damper,

DDE. Dynamic Data Exchange

DDE is primary a Windows-based mechanism that allows one application to send commands to another application, perform actions in that application, and have data returned. In most cases, one program (the client) is requesting data and the other (the server) is providing the data. For example, with DDE, you can insert a spreadsheet into a document created with a word processor and have changes made in the spreadsheet program automatically update the document.

Each data item that a server application can provide has a unique identifier consisting of three parts: an application name, a topic, and a DDE item name. The application name is typically the filename of the executable application running on the server. The topic is generally a category of data in the server application and the data item is uniquely identified by some means.

DDE links are always initiated by the client application, which broadcasts a message containing the unique DDE identifier to all other applications currently running. If a server application recognizes the DDE identifier and can provide the requested data it sends a response to the operating system, which establishes a link between the two applications. The link remains made until either application

fan, and others.

Containers allow users to organize the station database using a hierarchical structure much like a file system. Containers can be used in the same fashion that folders are used to effectively group control system components. There are basic containers, service containers, Web page containers, and composite objects.

Composite objects are used to encapsulate a group of nodes into a single, cohesive new object. The component parts of a composite object (child nodes) can be other primitive objects or composites themselves. Select inputs and outputs of the child nodes can be exposed directly on the composite. This allows the user to package complex control applications modularly and have them visually represented as simple graphical objects.

Lastly, notification services include notification classes that are used to route service messages and alarms to various user interface devices. In addition, this category includes mail recipient objects for routing alerts and alarms to various e-mail addresses.

JACE Controller Initialization. The JACE controller utilizes two types of memory for its basic operation. It uses dynamic RAM as main memory where all application code and data is copied for program execution while the station is running. The amount of main memory is crucial because that determines how many applications can run concurrently and how much data is readily accessible at any given time.

In addition to dynamic RAM, the JACE utilizes flash memory. In general, flash is used as non-volatile storage space for the basic operating system (boot flash), for storing application code that is necessary for object initialization and operation, and for storing object information that must live beyond the life of the object instance.

The station database is loaded into dynamic RAM from flash memory during start-up in a process that begins with C/C++ program code initializing its operating environment, which it establishes as a virtual machine separate from that of the underlying operating system. Once the operating environment has been established the station initializes the sub-systems that support it and loads the station database into PRISM. Each of the required sub-systems is loaded only once, then used by as many nodes as need it. With the station database loaded into memory, the station walks through the registry and loads the services it needs. A hash table is used to index these services. This is a highly efficient and common method of accessing data records by unique numeric index rather than searching through each record until the desired one is found.

The next step in the initialization process is for the station to initialize its nodes. The nodes register themselves with the services they need and the station kicks off an appropriate number of threads, then opens sockets for messaging. Once the initialization process is complete, the Web server can start and the

terminates it, the user selectively terminates it, or one of the applications is closed.

DDE client services in the Niagara Framework can establish both persistent and temporary connections between clients and servers. Once the client registers its interest with the server, it can read data from and write data to the server. When a persistent connection is established, the client continues to receive asynchronous updates whenever values change within the subsystem being controlled by the DDE server. Alternatively, the client can poll for data from the server using individual requests. When doing so, the link that is established between the client and the server disappears as soon as the data is retrieved.

IIOP. Internet Inter-ORB Protocol

IIOP is an object oriented protocol that is a critical part of CORBA, enabling distributed programs written in different programming languages to communicate over the Internet. Early versions of CORBA did not specify an ORB-internal communications protocol, which resulted in each ORB vendor determining how they would communicate internally. For that reason, the early versions of CORBA did not allow messaging between objects living in different ORB server processes. Consequently, CORBA 2.0 defined IIOP as a standard protocol that allows

user can log-on.

Note: On PC-based stations that support access to a physical disk drive (rather than flash memory), the initialization process is equivalent to that of the JACE controller except that persistence operations utilize the hard drive.

Station Platforms

The Niagara Framework is very scalable, which is an important feature because it allows end-users to invest in an architecture that they will not outgrow and one that easily integrates the same powerful functionality for configurations of every size. On a small building, a single JACE controller can be used to support a network of BACnet and LonWorks devices that can be accessed directly over the Ethernet LAN or remotely over the Internet.

On larger buildings, multiple JACE controllers can be connected on the Ethernet network to a JACE-NP (Network Processor) that provides multi-station supervision and multi-user access to the control system through either local or remote connection.

On multi-building complexes and large-scale control system integrations, the Niagara Web Supervisor manages global control functions, supports data passing over multiple networks, and hosts multiple, simultaneous client workstations connected over the local network, the Internet, or dial-up access.

It is important to point out that the same technologies, in fact, the same tools and services that are used to configure and monitor the smaller system are those that are available to support multi-user, multi-site configurations.

JACE Controller. The JACE controller is a network computer platform that supports embedded systems – specifically, the ChorusOS real-time kernel of JavaOS for Consumers. The JACE controller can distribute real-time control functions across an Ethernet LAN and run control applications stand-alone connected either locally to a JACE-NP or to a remote Niagara Web Supervisor over the Internet.

The JACE controller can be configured with or without Web UI services enabled. With Web UI services enabled, the station supports unlimited users over a secure intranet or remotely connected over the Internet via a standard Web browser.

Niagara Web Supervisor. The Niagara Web Supervisor is a turnkey control system and development environment whose minimum configuration includes a Pentium II Processor (400MHz or higher), 128Mb RAM, a one Gbyte hard disk, and Windows NT 4.0 (service pack 3 or higher). It is a desktop server platform that integrates the Niagara station with the relational database and application development environment (WorkPlace Pro). As a supervision station, the Web Supervisor includes the object database, the real-time control engine, and Web UI services, which provide user authorization for secure Internet access. In

communications between objects living in different ORB environments.

OLE. Object Linking and Embedding

Similar to DDE, OLE allows you to create objects with one application and embed them in another application. However, OLE provides greater control over shared data, therefore, it tends to be more efficient.

A key advantage of using OLE is that program control is temporarily transferred to the client application for the purpose of manipulating shared data. Whereas, with DDE, control always remains with the server application. OLE actually starts the source application when program control is transferred to the object and allows the user to edit data in its native environment. DDE does not provide this functionality because it can only activate other applications through Access Basic.

Support for OLE is built into Windows and the Macintosh operating system. A competing standard developed jointly by IBM, Apple, and Lotus is called OpenDoc.

RMI. Remote Method Invocation

An ORB (Object Request Broker) is the middleware that enables a client object to invoke a method on a server object. The ORB intercepts the call and becomes

addition, the Web Supervisor includes licenses that enable connection of multiple JACE controllers and/or WorkPlace Pro workstations.

As part of its duties, the Web Supervisor:

- Handles execution of the station database.
- Supports multiple, simultaneous client workstations over Ethernet, the Internet, or dial-up access.
- Provides database administration for the object database as well as relational data.
- Manages global control functions including global time synchronization and central scheduling, energy management, alarm processing and routing, and trend and data collection.
- Supports global data passing over multiple networks.

As a full-function engineering environment, the Niagara Web Supervisor includes a comprehensive application configuration tool set. It provides a graphic environment in which the developer can create, configure, and fully test control system application logic and user interface screens using soft-wiring techniques.

The tool set is complete in that it includes integral network management allowing for bindings to be made between LonWorks devices, BACnet devices, and existing systems. Through a simple linking mechanism, the user can share data between devices of different protocols transparently.

The Web Supervisor supports the latest Web standards including HTML 4.0, HTTP 1.1, XML, and cascaded style sheets. User access through the Web is supported on Microsoft Internet Explorer version 4.X and Netscape Communicator version 4.X.

JACE-NP (Network Processor). The JACE-NP is the middle-position compromise between the JACE controller and the Niagara Web Supervisor. It is small enough to provide a cost-effective solution for medium-sized control applications, but hefty enough to support multi-station operations. The JACE-NP is a compact PC platform with an integral hard disk, but no keyboard or monitor. It provides integrated, multi-station supervision and relational database management, but it does not include the WorkPlace Pro engineering environment. The JACE-NP supports distributed control applications over the Internet with your choice of standard browser access and/or third party user interface support.

responsible for finding an object that can implement the request, pass it parameters, invoke its method, and return results. RMI is the built-in native ORB that is packaged with JDK version 1.1.

Although RMI supports making method invocations on remote objects, it is not a CORBA-compliant ORB – it is native to Java. As such, RMI is a very natural mechanism for Java programmers to use because they do not have to leave the Java environment. However, this makes the use of RMI impractical for use with objects or application written in any other language.

WWW and Internet Related Standards

The terms Internet and World Wide Web are often used interchangeably, but they are not synonymous.

The Internet is a global network of computer networks connecting millions of users worldwide using a simple standard addressing system and communications protocol.

The WWW (World Wide Web) is a subset of the Internet. It is an information system composed of hypertext documents that are distributed over the Internet.

The following standards provide support for exchanging information on the Internet and the discussions include CGI, HTML, HTTP, MIME, and XML.

The Evolution of the Niagara Component Architecture

Much of the impetus for the design of the Niagara Framework has evolved from a revolution in computing technologies that started in the 70's with the popularity of the mainframe computer. At that time, IBM and Digital Equipment Corporation dominated the computing market. Mainframe computers provided large organizations with massive information processing capacity, but that capacity came with a very hefty price tag. The cost of entry could reach into the millions of dollars.

The mainframe model consists of an array of dumb terminals connected to one or more centralized computers over a network. The terminals are typically low-cost and consist of an ASCII display terminal, keyboard, and communications port. The network connection was initially a hardwired one, but more recently these connections are made via some form of telecommunications resource.

The advantages of mainframe systems include high performance, scalability, centralized management, low-cost desktop accessibility, and a high level of security. On the down side, mainframes are very expensive, typically proprietary, their applications are character-based, and as a single-point solution there is always the potential for bottlenecks and failures.

The minicomputer phase of business computing lowered the cost of entry to some degree, but it was PC LAN computing that significantly changed things in the corporate world. The PC introduced users to the graphical user interface and, along with thousands of Windows-based software applications, provided new levels of productivity for employees at every level of the organization. Low-cost PCs and powerful applications allowed organizations to extend their computing power out to the desktop, which reduced the dependence and load on centrally located mainframes.

Unfortunately, there is a premium to be paid for the power, flexibility, and increased productivity of distributed computing. Companies cannot afford to keep every desktop station upgraded to the latest technology, which is what is required to maximize the productivity of a distributed computing environment. This is what lead to the development of client/server solutions.

In a true client/server configuration, application processing is partitioned across both client and server. For example, in an SQL environment the client issues a query across the network to the server. The server hosts the database and processes the query on behalf of the client. The server then responds back across the network with the results of the query. This is in contrast to more traditional PC LAN computing configurations in which data itself, not just the query and the results, is transmitted over the network.

CGI. Common Gateway Interface

CGI is a specification that describes how a Web server communicates with a CGI program. A CGI program is typically a small program that resides on a Web server and processes data requests made of the server. A common application of CGI programs is to format user-entered form data into a database query. This type of program is known as a server-side solution because the processing occurs on the Web server (versus the client workstation). CGI programs can be written in any programming language, including C, Perl, Java, and Visual Basic.

A problem with CGI is that each time a CGI script is executed, a new process is started on the Web server. This can overburden the server and result in slow processing.

As an alternative to CGI programs, it is becoming increasingly popular to provide dynamic feedback to Web users by including scripts that run on the user's machine rather than on the Web server. Scripts are client-side solutions and are often implemented as Java applets, Java scripts, or ActiveX controls.

HTML. HyperText Markup Language

Although HTML derives from SGML, it is not a strict subset. SGML stands for Standard

Client/server computing begins to tap into the real potential of high-end centralized computing where the goal is to reduce the cost and maintenance of PC LAN configurations, provide powerful centralized computing capacity, while reducing network bandwidth requirements. Much like the mainframe model before it, client/server computing is shifting the responsibility of processing away from the desktop back to high-powered, highly reliable, multi-tiered centralized systems.

Client/Server Applications

Traditional client/server applications are two-tiered. The user interface runs on the client and the database is stored on the server. The actual application logic can run on either the client or the server. In what is known as *fat* or *heavyweight client* architecture, the client contains the presentation logic (window or display), the control logic (algorithms), and the data access logic (database connectivity). The server is relegated to managing the database. In *thin client* architectures, the client contains only the presentation logic. The control logic and data manipulation logic are partitioned into separate programs and deployed on one or more servers.

There are a variety of problems associated with two-tiered architectures. First, they impose significant administrative overhead because parts of the application must be installed on each client. Changes in the application logic can result in expensive upgrades for all client machines. Second, access is restricted to a set of proprietary client machines on which the application software has been installed. Next, scalability is restricted to the capacity of the server. As the number of clients increase, server performance suffers if its computing capacity is not upgraded accordingly.

Multi-Tier Applications

Multi-tier applications are partitioned into multiple distributed application components that are deployed as separate processes. Typically, the architecture consists of three tiers. As with traditional client/server applications, the user interface runs on a thin client and the database management system resides on a remote database server. What makes the three-tier model unique is that the control logic and the data access logic are partitioned and deployed on one or more servers separate from both the presentation logic and the database. The middle tier, or application tier, uses objects to interact with the two tiers above and below it – it processes requests from clients and manages database transactions.

The advantages of partitioning application components in this way include increases in performance, scalability, reliability, and manageability. In addition, multi-tier applications improve integration, they support multi-client access, they are highly flexible, and they provide a level of security that is not available

Generalized Markup Language, which defines a non-proprietary set of codes and conventions used in the printing industry to format printed documents. HTML, on the other hand, defines a similar set of codes that Web browsers use to format Web pages.

HTML codes affect how a Web page is displayed when viewed with a Web browser and they also indicate to the browser where to find other resources such as graphic images.

HTML formatting codes, or tags, are often called containers because they always come in pairs and they affect the text contained between them. A browser that encounters the tag **<B>** would display text in bold until it encountered the tag **</B>**, which turns the bold feature off.

HTTP. HyperText Transfer Protocol

HTTP is the network protocol of the World Wide Web. It defines the set of rules that applications use to exchange virtually all forms of data on the Web including HTML files, text, graphic images, sound, video, and other multimedia content. HTTP not only defines how messages are formatted and transmitted, but it also defines what actions Web servers and browsers take in response to various commands. A browser is an HTTP client, which sends requests to an HTTP server, which then sends responses to the client. For instance,

with heavy client architectures.

Performance. Moving the application logic to a separate server allows an application to take advantage of the power of multi-threaded and multi-processing systems. By leveraging the resources of multiple servers, server components have more ready access to processes, threads, database connections, and so on.

Scalability. As system demands increase, highly active components can be replicated and distributed across multiple servers to boost performance and support additional users.

Reliability. By replicating and distributing high-use components, single points of failure are avoided and bottlenecks eliminated.

Manageability. Thin client applications are easier and less costly to maintain. Very little code is actually deployed on the client machine. Most of the application logic is deployed and managed on the server. Fixes, enhancements, and extensions are all administered through centralized services.

Integration. The underlying theme of componentization is reuse and seamless integration. A function can be implemented once and reused in any application that needs it. Developers can pull from a library of components the functions they need and quickly assemble dependable applications.

Multi-Client Support. Any number of client applications can share the same application logic through its published interface.

Flexibility. The majority of application logic is encapsulated in relatively small, modular components that are accessed through a well-defined interface. As such, the component parts of an application can be changed without impacting other components inside or outside of the container. As a result, multi-tier applications can easily adapt to changing needs.

Security and Software Protection. Sensitive application logic often includes proprietary designs that could be reverse-engineered if deployed on the user's workstation. In addition, by leaving the logic on the server, user access can be controlled dynamically and revoked at any time.

Niagara Framework Components

The adoption of the Internet Protocol (IP) has provided a clear standard for digital communications. The number of physical media supported by IP has helped to provide a media-independent and scalable architecture that can support the range of requirements for applications spanning residential, commercial, and industrial markets. In addition, as interest in Web-based computing grows, there continues to be pressure put on manufacturers to move towards multi-tier approaches that require thin client architectures to support browser-based clients and rapid download of applets. Multi-tier architectures provide this thin client

when you enter a URL in your Web browser (http://www.sample.com), this becomes the command that directs the Web server to fetch and transmit the requested Web page, which may in turn contain references (hyperlinks) to other files whose selection will elicit additional transfer requests.

Usually, HTTP takes place through TCP/IP sockets. TCP/IP is the suite of communications protocols that are used to connect hosts on the Internet and sockets are the software objects that connect applications to those network protocols. In other words, when an application needs to send or receive TCP/IP messages, it simply opens a socket and reads data from or writes data to the socket – the operating system takes care of actually transporting messages across the network.

One unfortunate characteristic of HTTP is that it is stateless – it executes each command independently, without any knowledge of commands it may have executed previously. This has made it difficult to implement smart Web sites that react intelligently to user input. But, as all standards do, HTTP continues to evolve and HTTP 1.1 promises to address new needs and the shortcomings of HTTP 1.0.

Features of HTTP 1.1 include:

- Faster response, by allowing multiple transactions to take place

approach by removing much of the application code from the client and placing virtually all of the responsibility for life cycle and state management in the hands of the application tier.

The Niagara Framework is designed on just this type of architecture – multi-tiered and distributed. The Framework consists of three tiers: the user interface, the application tier, and the station database.

User Interface. As an Internet-based system, the Web browser has an important place in the Niagara Framework. However, browsers are not typically well suited for power users, nor are they outfitted to monitor critical alarms. As a result, the Niagara Framework provides both thin client and heavyweight client options.

Browser User Interface (BUI). The thin client option, or BUI, supports viewing the control system from any standard Web browser including Microsoft Internet Explorer version 4.X and Netscape Communicator version 4.X.

Niagara WorkPlace Pro. The heavyweight client, known as Niagara WorkPlace Pro, is a Java Swing-based graphical user interface designed as a cross between Windows Explorer and a Web browser. It includes views that allow browser-like viewing of the control system as well as a complete set of desktop tools to facilitate unique object-oriented distributed control application development and system administration. WorkPlace Pro supports multiple views that allow the user to graphically wire components when building application logic and user interfaces, it allows the user to view and test control logic in real-time, and it includes a programming language that can be used to customize components.

In addition, WorkPlace Pro features:

- An on-line session that allows the user to connect to a live running station over an IP network.
- An off-line session that allows the user to work off-line by accessing a station database directly.
- An application library, which is disk-based, accessible either locally or over a network, that contains resources for constructing a station database from pre-defined nodes, images, and composite control applications.

Application Tier. The application tier makes up the heart of the Niagara Framework. It includes a set of Java programs that extend the operating system to provide the object management services required to support distributed control applications over the Internet and a common application development environment and shared object services that fully support WorkPlace Pro. In addition, the application tier provides embedded network management that supports both BACnet and LonWorks devices.

The application tier can support multiple stations and multiple

over a single persistent connection.

- Faster response and great bandwidth savings, by adding cache support.
- Faster response for pages that are generated dynamically, by supporting chunked encoding, which allows a response to be sent before its total length is known.
- Efficient use of IP addresses, by allowing multiple domains to be served from a single IP address.

MIME. Multipurpose Internet Mail Extensions

MIME is a specification that defines the rules used to format non-ASCII messages so they can be sent over the Internet. Most e-mail clients now support MIME, which enables them to send and receive messages via the Internet with attached files that contain graphics, audio, and video.

In addition to e-mail applications, Web browsers support various MIME types. This enables the browser to display or output files that are not in HTML format.

XML. eXtensible Markup Language

XML defines a universal standard for storing and transmitting data. It provides a structural representation of data that can be used to

clients on appropriate PC-based server platforms (NT and Solaris) or relatively small, embedded solutions. It provides foundation services and a component control engine, multi-protocol interoperability, an extensive library of applications and protocol drivers, Web access services, alarm services, data collection and reporting, and much more.

Station Database. As previously discussed, the Niagara station runs within a JVM that provides an embedded, controller-level environment in which to configure, manage, and run the nodes and services required by a control system application.

In addition to the default .DB format, configuration databases can be exported in whole or in part to either XML format or SNS format. The DBADMIN database conversion program is used to convert the configuration database. In SNS format, a set of configured nodes is tightly packed into a binary stream for full or partial backup, storing application snippets in a library, for efficient network upload and download, and efficient flash memory storage. Also, the export and import features of the DBADMIN utility can be used to convert the configuration database to and from XML format to provide a standard open format for easy exchange.

Enabling Technologies

The Niagara Framework is an innovative software solution that allows mainstream technologies to be applied to software applications in the form of familiar user interfaces (Web browsers), rapid application development (Java), and ubiquitous network access (TCP/IP). Furthermore, the evolution of the Internet has made possible universal desktop access to vast quantities of diverse information including real-time data, real-time control, historical data, graphics, animation, and any combination thereof. Web browsers have provided both experts and novices with an inexpensive, common, powerful, and intuitive user interface for retrieving and navigating on-line information. Corporations are publishing internal data on secure Intranets and they have installed Web browsers as a standard desktop component and interface for accessing that information.

The Java Advantage

Java is a high-level programming language and environment designed by JavaSoft, a subsidiary of Sun Microsystems, that evolved out of a large-scale project originally tasked to develop advanced software for consumer electronics. The project set out to develop embedded solutions using C++, but a number of problems drove the development team to improve the language.

Since its beginning in 1995, the Java language continues to distinguish itself as the new standard for fully portable Internet applications and, as a multi-threaded solution, Java provides significant interactive responsiveness in real-time control environments. A number of Java's features make it well suited as

encode the content, meaning, and structure for any type of data from the simple to the complex. XML is similar to SGML and HTML in that it is a markup language, but its focus is defining the content itself, not presentation technique.

Like both SGML and HTML, an XML document holds text annotated by tags. With SGML and HTML these tags are predefined and they specify how documents are printed and displayed. However, with XML, there are no predefined tags. You define tags to suit your particular needs. Furthermore, the tags that you define tell programs processing the document how to understand the data. As a result, XML documents can be processed in ways that are impossible for SGML and HTML documents.

Lars Marius Garshol, in his *Introduction to XML*, presents an excellent illustration of how XML could be used. Garshol speculates that you could mark up recipes for simple meals with a DTD (Document Type Definition) that is tailored to accept amounts of each ingredient and alternatives for some of the ingredients. You could then easily create a program that, given a list of the contents in your refrigerator, go through the entire list of recipes and make a list of the dishes you could make that day. Alternatively, the program could sort suggested dishes by nutritional information, how long they would take to prepare, price, or any number

the foundation of the Niagara Framework.

Java is Simple. Java remains an object-oriented language very similar to C++, but is simplified to eliminate the compiler technology problems the design team encountered.

A common source of complexity in many C++ applications is managing memory. Java features automatic garbage collection, which is the periodic freeing of memory not being referenced. This makes the programming task easier, quicker, and it dramatically cuts down on bugs.

Another aspect of being simple is being small. Small Java software modules called Java applets can be downloaded from a Web server and executed locally on a client machine running any standard Web browser such as Netscape Communicator or Internet Explorer.

Java is Network-Savvy. Java includes a wealth of routines that have been validated with TCP/IP protocols like HTTP and FTP. This makes creating network connections easy – Java applications simply open and access objects over the Internet using URLs with the same ease that programmers use when accessing a local file system.

Java is Robust. As a compile-time language, Java provides extensive error checking at the time the program is compiled so that bugs are found before the application is distributed. Moreover, where C++ uses memory pointers, Java uses a memory address model that eliminates the possibility of overwriting memory and corrupting data.

Java is Secure. Java incorporates authentication techniques that are based on public-key encryption, which makes it a highly secure environment in which to work for distributed systems.

Java is Platform Independent. Java source code is compiled into a format known as bytecode that is easily interpreted on virtually any machine. This makes Java applications platform independent because they run the same in any JVM regardless of the hardware and software underlying the system. In addition, since the JVM has no direct contact with the operating system, there is little possibility of a Java applet damaging other files or applications.

Java is Multithreaded. Multithreading is a way of building applications that deal with many concurrent events. The interactive responsiveness of a multithreaded system is better suited for real-time environments than is a single-threaded system.

The Java Security Model

The key advantage of Java is that it was designed for use in distributed environments. Its authentication techniques make it unique for deploying secure applications across the Internet. The

of other criteria.

Enterprise-Level Standards

In the computer industry, the term enterprise is often used to describe the entirety of the computing resources in place to support a global organization. An intranet is a good example of an enterprise computing system. The following standards provide support for sharing enterprise-level information and the discussions include Ethernet, JDBC, ODBC, OSI, and TCP/IP.

Ethernet.

Ethernet is one of the first local area network protocols, developed by the Xerox Corporation in cooperation with DEC and Intel in 1976. Together, they created a 10 Megabit per second system, which later became the IEEE 802.3 communications standard. Since that time, 802.3 has been expanded to include a number of newer cabling options with the common goal of being simple, low-cost, compatible, high-speed, flexible, and maintainable. Ethernet is currently one of the most widely installed and accepted LAN choices worldwide.

IEEE 802.3 defines a communications standard that is a multi-access, packet switching network where each node has equal access to the network. Data is transmitted on the network and each node determines whether the

Java security model includes intrinsic security that protects the end-user from both intentional and unintentional errors. Unlike ActiveX controls, which have full access to the operating system and its resources, Java's Security Manager provides resource-level security that restricts a program's access to the disk, to the network, and so on. Although ActiveX controls tend to exercise more direct control than Java applets, this characteristic makes them more platform-dependent and they can compromise the integrity of the application that they support and its data.

Java programs are interpreted by the JVM and do not execute native machine instructions. Since they do not access system resources directly, do not modify or install new operating system or shared files, and do not require permanent installation on disk, they are far less likely to crash a system, breach security, or otherwise adversely affect system configuration and performance.

Finally, the Java security model introduces digital signatures that are assigned to applets to provide a means for extending the accessibility of software components from a trusted source. The digital signature verifies that the applet comes unchanged from its host, so the end-user can download and use it safe in the knowledge that the source code has not been tampered with either by intentional means or through some arbitrary programmatic failure.

Extending the Framework

The ultimate goal of the Niagara Framework is to build a technical foundation for next-generation, multi-vendor interoperable control systems that satisfy the need for Internet openness, while maintaining the determinism and integrity required for distributed control. Secondly, to help alliance partners maintain the uniqueness of their vertical market products while sharing a common object model infrastructure of distributed control functions.

The Niagara Framework enables rapid application development by providing a core set of reusable software constructs that are immediately useful for information and automation system applications. They provide building blocks that can be assembled into working control system components that comprise a complete information automation solution.

In addition to Niagara's capacity for rapid application development, the Framework provides the means to integrate an array of legacy, multi-vendor equipment under the umbrella of a single, integrated system. Niagara allows Facilities Managers to adopt new technologies and to integrate the latest in control system equipment and have these applications coexist seamlessly with existing equipment – without the use of inefficient and inflexible gateways.

Lastly, Niagara allows creation of custom control objects that

data is appropriate for it or not. The standard provides a CSMA/CD (Carrier Sense Multiple Access/Collision Detect) Media Access Control protocol, which means that any node on the network can transmit at any time, provided the network is quiet. After determining that the network is quiet, the node transmits its message. Should two nodes transmit simultaneously and a collision occurs, the network manager directs the nodes to wait a random amount of time, then retransmit.

Ethernet uses either baseband (one signal at a time) or broadband (multiple signals simultaneously) transmission. A variety of media can be used including 10BASE-T (twisted pair), 10BASE5 (also known as Thick Net, or the original spec using heavy coaxial cable, RG-11), 10BASE2 (also known as Thin Net, or a newer, lighter coaxial cable, RG-58), or fiber optic cable.

IEEE 802.3 also provides:

- Installation will support a bus or star topology.
- Data transfer rates of 10 Mbps or higher.

  A newer version of Ethernet, known as 100Base-T or Fast Ethernet, supports data transfer rates of 100 Mbps. The newest version, Gigabit Ethernet, supports data rates of 1 gigabit (1,000 megabits) per second.

easily integrate with the core set of constructs to extend the flexibility of the Framework. Program Objects provide compact software components that allow the user to customize and otherwise extend standard control solutions for their specific needs. Program Objects can have their variables bound to other objects without affecting the integrity of the application.

Sample Configurations

Figures 1-3 in the next section illustrate sample configurations that are representative of small, medium, and large-scale applications of the Niagara Framework. In its minimum configuration, the architecture supports a control system for a single, small building running a network of BACnet and LonWorks devices connected to a single JACE controller on the same field bus. Embedded Web UI services allow the JACE controller to support countless users over the intranet/Internet.

Figure 1 illustrates the use of the JACE-NP to support multiple stations on a single Ethernet backbone. As needed, the JACE-NP can be configured to interface the automation system to the information domain over the enterprise LAN. Local and remote browser user interface is supported by Web UI services embedded in the JACE-NP, which also provides station supervision for multiple JACE controllers and their network of LonWorks application devices.

Figure 2 illustrates a large-scale configuration where the Niagara Web Supervisor provides multi-client, multi-station support and integration of:

- Next-generation, multi-vendor, interoperable LonWorks control components over a LonWorks field bus.
- Utility metering over an optional RF link.
- Existing control system products connected over an RS-232 or RS-485/422 communications bus, directly connected to the Ethernet LAN, or by way of a DDE link through the Niagara Web Supervisor.
- Industrial process control networks.

Again, embedded Web UI services provide support for multi-user, multi-platform access to the control system from any intranet/Internet connection. In addition, the Niagara Web Supervisor includes licenses that enable connection of multiple JACE controllers and/or WorkPlace Pro workstations.

Figure 3 illustrates how the Niagara Framework can be used to support multiple buildings from a centrally located corporate server over the Internet. Local browser user interface is supported directly over the Ethernet LAN at locations where Web UI services are enabled on the JACE controller. In addition, these

- Ethernet is a bus contention system, not a token passing system.
- Supports a maximum of 1024 addressable nodes.
- Data is transmitted in packets including frames that define both source and destination addresses.
- Frame sizes can range from 64 bytes (minimum) to 1518 bytes (maximum).
- Ethernet defines the physical layer of the seven-layer OSI model, ensuring uniform physical and electrical characteristics.

JDBC. Java Database Connectivity

JDBC was developed by JavaSoft (Sun Microsystems) as a data access standard and it is the primary API for data access in the Java programming language. JDBC implements a standard SQL interface in Java classes, which enables Java applets to execute SQL statements. This allows Java applets to interact with any SQL-compliant database. Since virtually all relational database management systems (RDBMS) support SQL, and because Java itself runs on most platforms, JDBC makes it possible to write a single database application that can run on different platforms and interact with different RDBMSs.

JDBC is similar to ODBC, but

services enable high-speed, remote browser user interface over the Internet through a network of cable/ADSL modems. Lastly, remote system monitoring and support is provided over the Internet through the Niagara Web Supervisor.

it is designed specifically for Java applets, whereas ODBC is language-independent.

ODBC. Open DataBase Connectivity

ODBC was developed by Microsoft as a language-independent standard database access method to provide users with a mechanism to access any data from any application, regardless of which database management system (DBMS) is handling the data. ODBC employs a database driver inserted between the application and the DBMS to translate application database queries into commands that the DBMS understands. For ODBC to work, both the application and the DBMS must be ODBC-compliant.

OSI. Open Systems Interconnection.

See *Protocol Stack* in the Glossary.

TCP/IP. Transmission Control Protocol / Internet Protocol

TCP/IP is a set of protocols developed to allow computers to exchange information across a network. Most notably, TCP/IP has become the “Internet protocol suite” or the communications language of the Internet. Moreover, TCP/IP is now being used in private networks, intranets, and extranets for applications including e-mail, file transfer (FTP), and remote login (TELNET).

Customer Benefits

Tridium recognizes the value of open technologies that allow interoperability of multi-vendor equipment and compatibility of control and information management applications. The benefits of true openness are extensive, but the single most valuable result of adopting the Niagara Framework is that it puts System Integrators, Energy Services Providers, and end-users in control of their facilities, letting them choose the products they want, with the features they need, from the vendors who perform to their expectations.

The Niagara Framework is secure, scalable, cost effective, robust, and extensible. In addition,

- Niagara provides interoperability through a distributed object infrastructure and native support for standards. It is NOT just a gateway solution.
- The architecture promotes platform independent applications that are plug-and-play including BACnet, LonWorks, and legacy integration.
- The framework solution simplifies the automation and information architecture across the entire enterprise and can significantly reduce hardware, software, engineering, training, and support costs.
- Integrating automation data into business information systems is no longer a manual process.
- Niagara eliminates the need to replace versus integrate.
- End-users can efficiently reconfigure their control systems to fit changing circumstances, without entering into a lengthy customization process.
- The Niagara Framework allows alliance partners to maintain the uniqueness of their vertical market products, while sharing a common object model infrastructure of distributed control functions.
- As a complete solution, Niagara provides a seamless, networked environment that can quickly bring to market the precise products and services the customer demands.

The Niagara architecture makes it possible for anyone with appropriate security, from anywhere, at any time to interact with smart devices embedded in building, power/utility, and industrial control networks through a standard Internet browser.

TCP/IP combines the Internet Protocol (IP) at the lower end, which allows you to deposit and retrieve datagrams, and the Transmission Control Protocol (TCP) at the higher end, which allows you to establish a virtual connection between the source of the data transmission and its destination.

TCP manages the disassembly of messages from network applications into smaller packets that are handed to the IP layer for transmission over the Internet. It also manages the re-assembly of packets into the original message for processing by network applications. IP interfaces with the TCP module above it and most typically an Ethernet driver below it to address and route messages so they get to their destination.

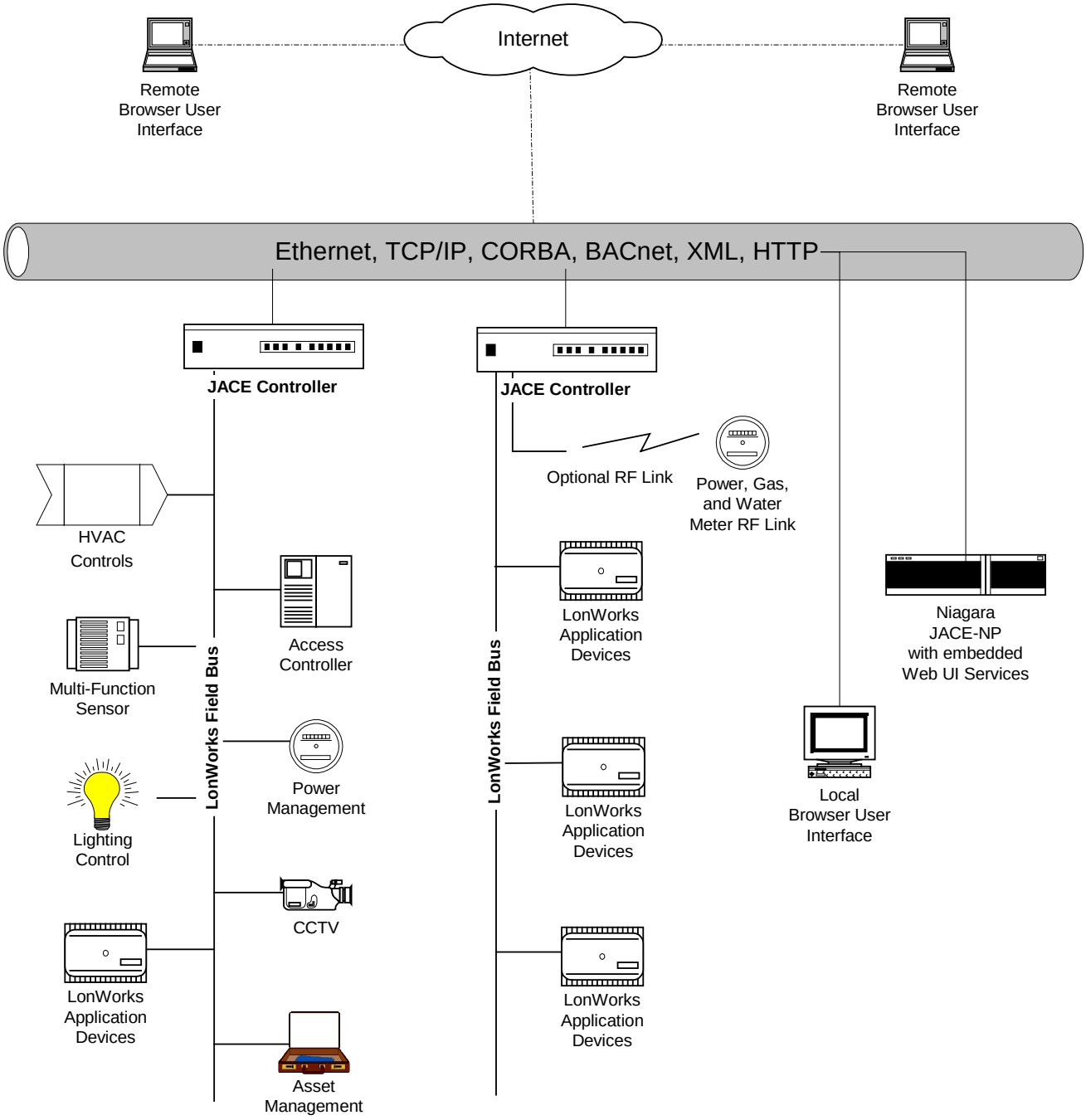

Figure 1.

JACE-NP Configuration

Local and remote BUI access to the control system network via Intranet/Internet, Ethernet, TCP/IP, CORBA, BACnet, LonWorks, and Optional RF Link. The JACE-NP provides multi-station support for mid-sized integrated BAS network.

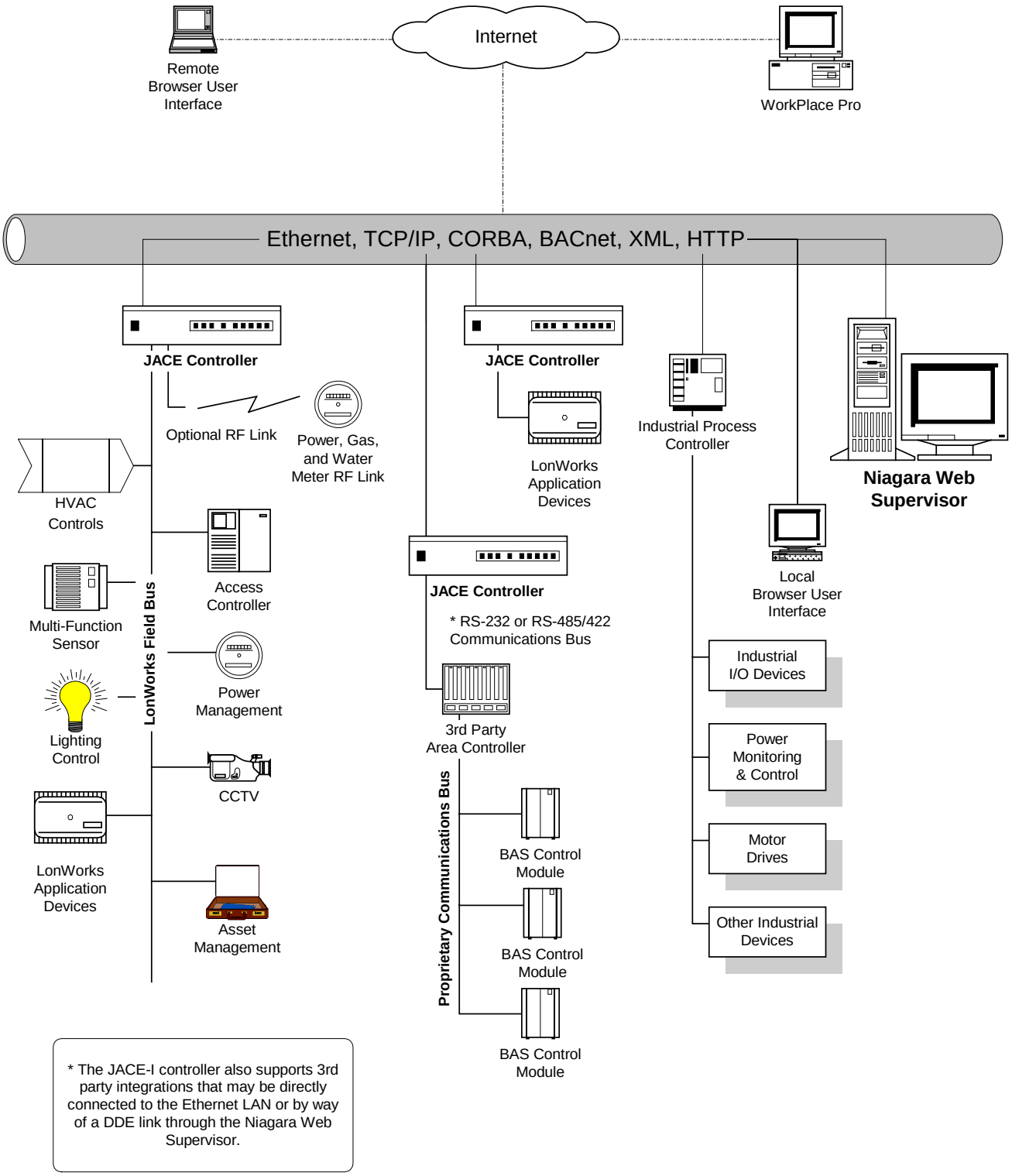

Figure 2.

Large-Scale, Web Supervisor Configuration

Local and remote BUI access to large-scale integrated BAS and process control networks via Intranet/Internet, Ethernet, TCP/IP, CORBA, BACnet, LonWorks, and proprietary communications protocols. Niagara Web Supervisor provides multi-station and multi-client support and hosts the WorkPlace Pro application development environment.

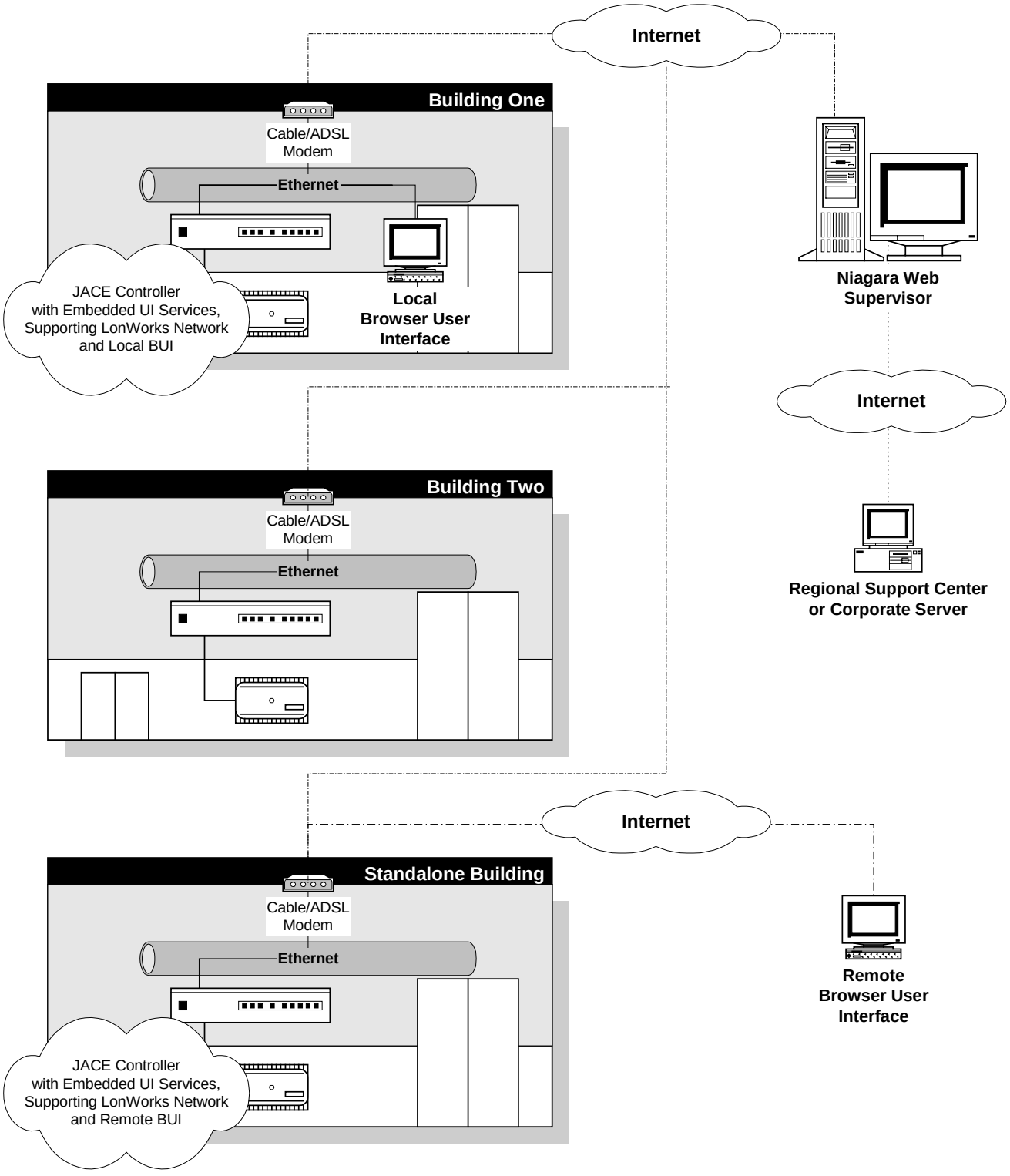

Figure 3.

Remote Small Building Monitoring & Control

Local and remote small building monitoring and control over Internet cable/ADSL modem network. Niagara Web Supervisor provides multi-station and multi-client support for local/regional support center or corporate server.

Glossary of Terms

ActiveX control

ActiveX controls can be automatically downloaded and executed by a Web browser and they provide functionality that is very similar to Java applets – they enable Web authors to embed interactive elements in their Web pages. ActiveX controls, however, have full access to the operating system (Windows). This tends to make them more powerful than Java applets, but this power erodes their portability and security. By being tied to Windows, ActiveX controls are not platform independent user interface controls and they can compromise the integrity of the application that they support and its data.

Programmers can develop ActiveX controls in a variety of languages, including C, C++, Visual Basic, and Java.

applet

A smaller application program designed to be executed from within another larger program. Comprehensive application programs are executed directly from the operating system. Applets, on the other hand, are smaller modules that can be called from a variety of applications. For instance, applets can be downloaded from a Web server, by a Web browser, and executed on a client machine. With the growing popularity of object-oriented programming and OLE (object linking and embedding), applets are becoming very popular.

application library

A collection of pre-built and pre-tested software components that perform specific functions. They are sometimes called modules or routines. The routines are stored in libraries, then accessed by the application programs that need them. This is particularly useful in that frequently used routines can be stored in a common pool, called by any number of applications, and updated as needed without directly impacting the parent application program. In Windows environments, library files have a .DLL extension.

ASHRAE

American Society of Heating, Refrigerating, and Air-Conditioning Engineers, Inc.

authentication

This is the process of identifying a user, typically by having them enter a username and a password as part of a log-in procedure during system start-up. Authentication merely identifies the user, but says nothing about the access rights of that person.

Authorization is different from authentication, in that authorization is the process of granting a user access to certain functions and objects based on their identity.

In addition to using *authentication* as it relates to password access, the term also has meaning in the LonWorks domain. As it relates to Echelon devices, authentication means one of two things. First, if authentication is enabled on an Echelon device, only network management tools using authentication messages can change that device's configuration. Next, network variables can be configured to accept only authenticated updates.

BACnet

Building Automation and Control Networks

See Sidebar.

cascaded style sheets (CSS)

In word processing and desktop publishing, style sheets define templates for page layout. They can be used to define page size and orientation, font size and style, paragraph spacing, tabs, borders, and more. Style sheets are very useful in defining common templates that can be used over and over for similar type of documents. For example, you can define style sheets for personal letters, professional reports, newsletters, and so on. As with desktop publishing, style sheets can be used in Web page design.

In considering the application of style sheets, one must consider that both publisher and reader may wish to influence style. Publishers may wish to have a distinctive look that they implement for all of their documents, where users may either wish or need to restrict the use of some elements based on system capacity or personal preference.

In an effort to accommodate both publisher and users, conflicts may arise. Whose style preference wins out when there is a conflict between the styles publishers provide and the styles users wish to use? CSS was developed by the W3C (World Wide Web Consortium) to resolve these conflicts. CSS includes a scheme for assigning priority to each style element: styles may be identified as “normal” or “important”. If two styles have the same priority, the publisher’s style sheet takes priority. Otherwise, the style with the highest priority wins. Note, however, that users can disable style sheets altogether (through browser settings), which give them ultimate authority over the user of style sheets.

CGI

Common Gateway Interface

See Sidebar.

client

One side of the client-server relationship. Client-server is a network architecture where one or more high-end computers (servers) process dedicated tasks and manage shared resources for a collection of PCs or workstations (clients) on which users run applications. The clients rely on the servers for various resources including drive space, files, printers, and even processing power.

CORBA

Common Object Request Broker Architecture

See Sidebar.

DCOM

Distributed Component Object Model

See Sidebar.

DDE

Dynamic Data Exchange

See Sidebar.

distributed control system

A system of dividing automation control into several areas of responsibility, each managed by its own controller or processor, with the whole interconnected to form a single entity usually by communications buses of various kinds.

enterprise information system (EIS)

The IS department for many companies is the department that is responsible for computers, networks, and information management. EIS refers to information systems management across an entire organization, which may be composed of several companies and locations (globally). An intranet is a good example of an enterprise computing system.

Ethernet

See Sidebar.

extranet

An intranet that is partially accessible to outside users. Typically, intranets reside behind firewalls and restrict access to inside users – those with connectivity on the secure side of the firewall. Extranets, though, provide various levels of access to authorized outside users.

Extranets are becoming a popular means for business partners to exchange information over the Internet.

field bus

Years ago, when process and automation control system architectures were highly hierarchical, there was a clear separation between the categories of communications layers. In the process world, the sensor bus supported discrete sensors and actuators, the device bus supported analog sensors and actuators, and the field bus delivered measurements via a 4-20mA loop. Above these was a supervisory bus that supported a data network of operator consoles.

In the BAS world, sensors connect directly to controllers, smart actuators and application-specific controllers connect on a controller trunk, and area controllers provide global control of the network. Above these, the user has the option of connecting workstations to gather control system data.

As control system architectures have flattened, there has been much confusion created associated with the use of the terms: sensor bus, device bus, field bus, and now information network. The growth of digital communications technology and the push for open communications protocols have led to a vast array of networking options that are eliminating the distinction between device-level communications, field bus networks, area controllers, and even that of the information network.

Field bus refers to an all-digital, serial, two-way communications architecture that includes sensors and actuators, smart actuators and controllers, stand-alone processors, and user interface appliances interconnected on a single, fully interoperable network. As such, the field bus serves as the Local Area Network for all devices beneath a JACE controller.

firewall

Firewalls are frequently used to prevent Web surfers from accessing networked resources connected to the Internet, especially intranets. Firewalls can prevent outside users from accessing a private network and they can block certain messaging from being placed on the Internet by inside users. They can be implemented in either hardware or software, but they are most often implemented as a combination of the two. All messages entering and leaving the intranet pass through the firewall, which examines each message and blocks those that do not meet certain criteria.

flash memory

Traditional non-volatile memory came in the form of erasable/programmable read-only memory (EPROM). The greatest disadvantage of EPROM technology was the cost and inconvenience associated with re-programming the device, which required an external UV light source. EPROM gave way to EEPROM, which is electrically erasable and imposed less of an interruption in system operation for the purpose of changing the code programmed therein.

More recently, flash memory devices have further reduced the overhead imposed by making code changes after the system is in the hands of the user. Like EEPROM, flash memory can be re-programmed while in the system, but it is significantly faster because data can be written and erased in blocks rather

than one byte at a time. Flash memory is solid-state, it has no moving parts, and it does not require a battery to retain data for over one hundred years. System power is needed only during a read or write cycle.

To this point, the focus of using flash technology has been on code storage with memory capacity requirements of less than 1 Mbyte. Historically, system designers used flash memory in the design of built-in software (BIOS) so it could be updated quickly and easily. But, continuing advances in memory technology are driving the use of flash devices beyond traditional non-volatile storage designs. Today, the flash memory market is diverging into products that support both code storage and data storage applications, where the memory capacities reach 32 Mbytes and beyond. With the data storage requirements of flash approaching those of dynamic RAM, performance improvements (speed and block protocols) are critical.

Unlike traditional non-volatile memory devices, which emulate ROM storage, the flash technology adopted by the Tridium design provides high-capacity, high-speed, embedded or removable mass storage that emulates disk-based storage. One Mbyte of boot flash and at least 32 Mbytes of serial flash are used on the JACE controller to provide highly efficient persistent storage of the station database. The database is loaded into dynamic RAM from flash during station start-up and copied back to flash on a regular basis to provide non-volatile backup. As an alternative to on-board flash, the JACE controller provides optional PCMCIA connectivity for removable mass storage.

hierarchical structure

A hierarchical structure describes a system whose organization resembles that of a tree. In other words, components of the structure are linked to multiple subordinate components beneath them. A hierarchical structure may reflect the organization of a file system, where directories contain sub-directories and sub-directories contain files. The menu structure of a program may be hierarchical in that the root menu provides access to multiple sub-menus below it.

Hierarchical structures have been very popular in the design of control systems. Much of the installed base consists of architectures with host processors supporting a network of area controllers each supporting a network of smart actuators and application-specific controllers to which are connected sensors and controlled devices.

There are a variety of problems with a hierarchical design in control system architecture. Because of the star-type topology, installation can be costly, throughput congested, and troubleshooting complex. Because central controllers manage communications between their nodes (and other controllers), a failure at the controller level can affect monitoring and control of a complete section of the system. As control devolves to the field bus level, system architectures are becoming more flat and there is a definite move away from hierarchical control system design.

In addition to these uses, the term hierarchical is used to describe the structure for storing nodes in the station database. The root of the database is a station node in which containers store various groupings of control system objects. These objects can be containers themselves, composite objects, individual control applications, groups of services, or whatever structure best suits the user.

HTML — **HyperText Markup Language**

See Sidebar.

HTTP — **HyperText Transfer Protocol**

See Sidebar.

hyperlink — An element contained on a Web page that links the user to a different Web page on the same site or an entirely different site. Hyperlinks are also used as user controls in electronic documents that take the user to another place on the current page or to a completely different document.

Typically, you click on a hyperlink to navigate a Web site, to link to related topics, or to display glossary pop-ups for hi-lited terms in electronic documents.

IIOP — **Internet Inter-ORB Protocol**

See Sidebar.

Internet — The Internet is a global network of computer networks connecting millions of users worldwide using a simple standard addressing system and communications protocol.

interoperability — The ability of multiple vendors' products to be integrated into one flexible system without using custom integration products. There are varying degrees of interoperability.

JavaBean — Sun Microsystems defines a JavaBean as "a reusable software component that can be manipulated visually in a builder tool."

Software component models provide conventions for writing user-interface controls that are platform-independent. In other words, component models strive to define a set of rules that provide for rapid development of reusable software modules so that application development tools and runtime environments running on various computer platforms can manipulate them without the need to recompile. Prior to the release of the JavaBean API Specification, there were two popular component architectures: Microsoft's ActiveX and a competing standard developed by IBM, Apple, and Lotus known as OpenDoc.

The 1.0 level of the JavaBeans specification came out in October 1996 and it became the foundation on which the Java Development Kit (JDK) was based in 1997. Unlike ActiveX controls, which are largely tied to the Windows operating system and distributed as machine-specific binaries, JavaBeans can run anywhere. The obvious trade-off is that JavaBeans may not be able to take full advantage of their local environment without compromising this portability. Nevertheless, the JavaBean component architecture becomes the obvious choice for developing components that can be downloaded from the Internet.

Java controller — Traditional proprietary automation controller platforms have had too little memory, speed, and throughput to support multiple building service applications. Therefore, each application such as HVAC and access control has been provided on separate controllers. This leads to a less comprehensive solution where installation costs are high and integration is difficult if not impossible.

These limitations can be overcome by employing Java controller processor technology, which lets you integrate multiple building service applications on a

common computing platform.

A Java controller is a network computer platform that supports embedded systems – specifically, the ChorusOS real-time kernel of JavaOS for Consumers. The Java controller, connected to an interoperable BACnet and LonWorks field bus, can distribute real-time control functions across the Ethernet bus, run control applications stand-alone, and provide users with Internet connectivity.

Not only is the Java controller extremely flexible in terms of providing comprehensive integration solutions, it is low-cost. The platform itself is competitive with other automation controllers, the installed cost is significantly reduced, and it can be administered and updated from a central network server.

Java Virtual Machine (JVM)

The Java virtual machine is a self-contained operating environment that behaves as if it were a separate computer. It acts as the bytecode interpreter and runtime environment for Java applets, which are not given any direct access to the host operating system. This design has two distinct advantages: system independence and security.

Java applications are platform independent because they run the same in any JVM regardless of the hardware and software underlying the system. In addition, because the JVM has no direct contact with the operating system, there is little possibility of a Java applet damaging other files or applications.

JDBC

Java Database Connectivity

See Sidebar.

kernel

The central module of an operating system. It is the part of the operating system that loads first and remains in main memory. Because it stays in memory, it is important for the kernel to be as small as possible, while still providing all the essential services required by other parts of the operating system and applications. Typically, the kernel is responsible for memory management, process and task management, and disk management.

LON

Local Operating Network

See Sidebar.

metadata

Metadata is data about data. It describes attributes of the data itself – it can describe how data is collected and stored, when it was collected and by whom, and it can describe how data is formatted.

The Niagara object model includes property descriptors that provide a much richer metadata model as compared to standard JavaBeans. For instance, Niagara property descriptors include information about security privileges, engineering units, and enumeration tags. In addition, property descriptors can cross-reference other properties (property X's engineering units are stored in property Y). With data typing refined to the primitive level, data access is very quick and performance is significantly enhanced for the distributed real-time environment.

MIME

Multipurpose Internet Mail Extensions

See Sidebar.

network management

In the context of intelligent control networks, network management refers to the tools and services that support the process of logically addressing nodes,

configuring them, and binding their network variables.

object-oriented programming (OOP)

Object-oriented programming is a technique that employs the use of software constructs that not only contain code (sequences of computer instructions), but also data (the information on which the code operates). These constructs are called objects and they tend to insulate the programmer from the functional details of the object by virtue of the way they operate. Everything an object can do is represented by its message interface. In other words, objects simply receive and send messages. The message that an object sends is determined by the message it receives and the methods it contains. Methods are the actions that objects carry out.

Objects are defined by their class. Objects are individual instances of a class and all that is needed to create a new class of object is to create a subclass of the original that most closely matches the behavior that is needed. The subclass inherits the entire behavior of the original and it allows customization. It is the reusability of objects (or their inheritance) that makes object-oriented programming so efficient. Programmers simply create new objects that inherit many of the features of an existing construct and they can be assured that it will function properly. This approach speeds the development of new programs, provides reusability, and improves maintenance.

There are a number of object-oriented programming languages in use today including C++, Smalltalk, and Java. C++ is the object-oriented version of C that uses a “compile-time binding” technique, which makes for high runtime efficiency and relative small code size, but it can lead to some inefficiencies in terms of reuse. Smalltalk, on the other hand, uses a “runtime binding” technique, which means that nothing about the object needs to be known until the program is run. As a result, Smalltalk programs are considered to be faster to develop than C++.

Java is the latest of the object-oriented programming languages and it is closely tied to the Internet and Web browser development. It also provides some important improvements over both C++ and Smalltalk. C++ uses memory address pointers, whereas Java uses a memory address model that eliminates the possibility of overwriting memory and corrupting data. Java uses automatic garbage collection, which is a feature that frees the programmer from having to explicitly allocate and de-allocate memory. Also, Java runs on a virtual machine, which is software built into the Web browser that executes the same Java bytecode no matter what type of computer platform is being used.

ODBC

Open DataBase Connectivity

See Sidebar.

OLE

Object Linking and Embedding

See Sidebar.

OSI

Open Systems Interconnection

See *Protocol Stack*

primitive

Primitive is a term that refers to subordination. In programming, primitives are the basic operations supported by the language from which procedures are built. Primitives can be low-level objects from which higher-level, more complex objects are constructed (composites).

Primitives can also refer to the properties that define an object, but more

importantly their component data types. In this case, primitives are integers, floating-point values, Boolean values, or other data that constitute the data store known as a property.

Also see *Metadata*.

protocol stack

Protocols define a common method for communications between computers. In general, a network protocol defines how communications should begin, how communications should end, and the sequence of events that occur in between. Protocols are established by standards committees then implemented by hardware and software manufacturers in their products.

At the transmitting computer, protocols break down data that is to be transmitted into smaller segments called packets. These smaller segments can be transmitted much quicker, resulting in significant response time improvements when large amounts of data are to be transmitted. Protocols are also responsible for adding addressing information and preparing the data for transmission through the network interface card (NIC) and the transmission media (wire, fiber, etc.).

At the destination computer, protocols are responsible for collecting packets, striping off formatting information, copying the data portions of the message to a memory buffer, then reassembling the message in the proper order and checking it for errors.

The work of several protocols must be coordinated to prepare, transfer, receive, and act upon data transmissions. This is referred to as layering. The OSI (Open Systems Interconnection) Reference Model is one such combination of protocols where each layer is responsible for handling a function or subsystem of the communications process. The OSI model defines seven layers:

Application Layer	Initiates or accepts a request.
Presentation Layer	Translates data into a format that is understood by the receiving computer, can compress or expand data, and can encrypt or decrypt data.
Session Layer	Establishes and manages transmissions between computers, synchronizes message transmissions, and communicates errors from upper layers.
Transport Layer	Organizes higher level messages, delivers segments to higher levels, and detects and attempts to correct problems in the network.
Network Layer	Determines best routing and adds sequencing and addressing information to the packet.
Data Link Layer	Establishes and maintains connections between computers, performs media access control, and deals with errors from upper layers.
Physical Layer	Carries messages between computers. Ethernet is one of the most widely installed and accepted implementations of the physical layer of the OSI model.

The various layers of protocols and the software that is used to process those protocols are often referred to as a stack. Programmers often talk about

loading a stack, which means to load the software required to use a set of protocols. Another common phrase is binding a stack, which refers to linking a set of network protocols to an NIC. Every NIC must have at least one stack linked to it, but if more than one stack is bound to a particular adapter, the binding order determines which protocols are used to attempt a successful connection.

real-time control system

A real-time control system responds in a timely and predictable way to unpredictable external events. The key characteristics of a real-time control system are:

- Timeliness. It must respond very quickly to applications, which are required to complete certain tasks within the time boundaries it has to respect.
- Simultaneity or simultaneous processing. Even if more than one event happens simultaneously, all deadlines must be met.
- Predictability. Also known as determinism, one characteristic of real-time control is that reactions to external stimuli are very predictable – all possible events are reacted to in precisely the same fashion every time.
- Dependability. It is necessary that the real-time system environment be ultimately reliable.

relational database

A relational database is a database that contains multiple tables that are related through common fields. This differs from a flat-file design where each database is self-contained in a single table. Relational databases are powerful because they require few assumptions about how the data is related, the number of tables and their size, and how data may be extracted from the database. The key advantage of a relational design is that data can be viewed in many different ways from a single database with virtually no duplication of data in the various tables that comprise the database.

The database management system associated with a relational database is referred to as a RDBMS and almost all full-scale database systems are of this type.

RMI

Remote Method Invocation

See Sidebar.

RTOS

Real-Time Operating System

It is important to distinguish between a real-time system and a real-time operating system. The real-time system represents the set of all system elements including the hardware, the operating system, and the applications that are needed to meet the system requirements. The RTOS is just one element of the complete real-time system and it must provide sufficient functionality to enable the entirety of the system to meet its requirements.

It is also important to distinguish between what are simply fast operating systems and true RTOSs. Speed, although useful for meeting the overall requirements of an RTOS, does not by itself meet the requirements of one.

A popular Internet newsgroup associated with RTOS design lists some of the requirements of RTOS to be:

- It must be multi-threaded and preemptive.
- It must support thread priority. A system of priority inheritance must exist

and it must support predictable thread synchronization mechanisms.

- The behavior of the operating system must be predictable. This means real-time system developers must have detailed information about the system interrupt levels, system calls, and timing.

The ChorusOS kernel provides the foundation for JavaOS for Consumers, the RTOS of the Java controller. ChorusOS provides the real-time functionality and determinism required for distributed control systems. Its modular architecture, with replaceable systems components allows ChorusOS to scale seamlessly from very small, embedded systems to high-functionality, transparently distributed platforms.

serialization

Object serialization is the process of transforming objects and their references into streams of bytes. This mechanism is not only effective for storing and retrieving objects to and from disk or flash memory, but also for communicating via sockets. By default, serialization writes and reads non-static and non-transient data fields, which makes it both fast and lightweight. In addition, it provides a certain level of security in that secure data need only be declared transient and therefore not serialized.

servlet

Where applets are small application programs that are executed from within another larger program, servlets are applets that run on a server. This term usually refers to a Java applet that runs within a Web server environment.

Java servlets are becoming an increasingly popular alternative to CGI programs, which typically process data requests of Web servers. The key advantage of Java servlets over CGI programs is that they are persistent. This means that once a Java servlet is started, it stays in memory and can fulfill multiple requests. The key advantage to persistence is that it makes Java servlets much faster because there is no time wasted in setting up and tearing down the process.

TCP/IP

Transmission Control Protocol/Internet Protocol

See Sidebar.

URL

Uniform Resource Locator

URLs provide a global addressing scheme for documents and other resources on the World Wide Web.

The first part of the address indicates what protocol is used. The second part specifies the IP address or the domain name where the resource is located, and the third part (optional) defines the page to retrieve. For example, the two URLs below point to two different files at the domain name tridium.com. The first specifies a document file that can be fetched using FTP protocol and the second specifies a Web page that can be fetched using HTTP protocol.

ftp://www.tridium.com/info.pdf

http://www.tridium.com/index.html

Another less widely used term that relates to global addressing of resources on the WWW is Uniform Resource Identifier (URI), which is a generic term referring to all types of names and addresses for objects on the WWW. A URL is one kind of URI.

WWW	**World Wide Web** The World Wide Web, or Web, is a distributed hypertext-based information system distributed worldwide on the Internet.
XML	**eXtensible Markup Language** See Sidebar.

REFERENCES

http://www.webopedia.internet.com/. Mecklermedia Corporation, 20 Ketchum St. Westport, CT. 06880 USA.

http://www.realtime-info.be/. Real-Time Consult, Rue de la Justice, 1070 Brussels (Anderlecht), Belgium.

http://www.sun.com/. Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, CA 94303 USA.

http://www.jmarshall.com/. © 1997 James Marshall.

http://omg.org/. Framingham Corporate Center, Object Management Group, Inc., 492 Old Connecticut Path Framingham, MA 01701 U.S.A.

http://fieldbus.net/. Datalapio Oy, P.O. Box 97, 00331 Helsinki, Finland.

http://www.weblogic.com/. BEA WebXpress, 550 California, San Francisco, California 94104.