ECHELON®

Considerations for Using XML Web Services for Device-to-device Communication

Alex Chervet
Marketing Manager

Introduction

Since the early 90's, the idea of interoperability and integration of automation systems has been the "Holy Grail" of the commercial building and industrial automation industries. Both committee-based and market-based solutions have emerged and gained significant market share with ANSI/EIA 709.1 networks now accounting for a significant portion of all new system installations.

Yet as the market for control networks developed, a new set of standards took root in the IT industry that forever changed information access. This, of course, was the advent of the World Wide Web.

Today, the Web has progressed from information source to information distributor. It has become a major vehicle for commerce and business. In fact, Web and IT technologies, in general, have become the stock answer to all that ails the automation industry. Some even argue that existing device-level technologies should be replaced with Web-based Ethernet networks. While this may make sense for higher-level gateway/supervisory functions and system-to-system integration, it is not the case at the device level.

In the same way that the Web disintermediated knowledge from its source, IP networking in the form of XML and web services, will disintermediate device networks from higher-level applications and enterprise knowledge systems.

This paper presents the unique needs of device networks and the Internet and emphasizes how, when melded together, these two technologies will provide the longest term value and greatest return to end-users, as well as providers of automation systems. The focus of this paper is on the commercial building automation industry.

What Do You Mean by "Device"?

For this discussion "device" means [inexpensive] simple sensors, controllers, and electro mechanical end effectors typically found in commercial buildings like damper or valve actuators, card readers, occupancy sensors, thermostats, and light sensors – not home automation devices like personal video recorders (PVRs), access points, stereo gear, or printers.

These devices are assembled into systems that typically take on the following architecture:

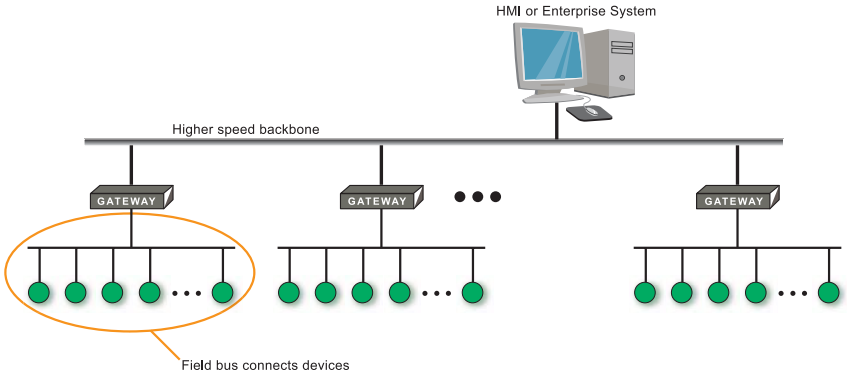

FIGURE 1 – Typical System Architecture

In systems like these, the devices tend to be the high-volume items. A building may have hundreds of occupancy/light sensors, but only a single lighting panel. An elevator bank may have thousands of push buttons, but only 8 or 10 cars. For these reasons, devices need to be inexpensive and easy to assemble into a system.

Installers of these devices do not typically work with computing tools such as laptops. They are often tradesmen such as journeyman electricians, or pipefitters. If a computing device such as a laptop computer or hand-held unit is required to commission the system, depending on the installer, this may require a "higher level" technician.

XML Web Service Challenges and Opportunity

Most web service implementations use HTTP, SOAP and XML over Ethernet networks. Other variations are possible, but this is by far the most popular implementation and the one best supported by today's tools. Thus, the most common and cost-effective way of implementing web services in a device would be to use HTTP over Ethernet. A typical example would be to add an RJ-45 jack to a thermostat and connect it to other devices via Ethernet.

Challenges

The challenges facing the adoption of XML web services at the device level fall into three major categories: system cost (including added device costs), management, and security.

SYSTEM COSTS

Device Cost

Today's tools support a web services "stack" usually consisting of TCP/IP, HTTP, SOAP and XML, as well as support utilities (XML parser, XSLT engine, XPath evaluator, etcetera). This "stack" (as well as the support utilities, if provided) requires more CPU horsepower and memory than is typically found in low-cost devices. Microcontrollers such as the Toshiba 3120/3150, Intel 8051/8032, Motorola HC11/HC05, or other small microcontrollers from Cypress, Microchip, Dallas, and others, simply do not have the computing power to support the infrastructure required to implement Web services, thus Web service enabled devices are likely to be built around "larger" processors or "larger" variants of those microcontrollers.

Since XML Web Services are usually transported over HTTP, often as text, each web service call requires more bandwidth than a compressed binary protocol, thus faster networks are required to transport web services.

Supporting TCP/IP, HTTP, SOAP, an XML Parser, an XSLT engine, and XPath evaluator requires more memory than is typically available in microcontrollers. This means solutions will require larger amounts of external memory. These three requirements conspire to substantially increase device cost, and, because systems consist mostly of devices, the multiplicative effect significantly increases the cost of the entire system.

Wiring

Control systems are wired differently than Ethernet networks, which are typically installed using a "home-run" or "star" topology. "Daisy-chain," loop or "free" topologies are often preferred to star topologies when wiring for control and attempting to lower installation/wiring costs for thousands of control points. After all, if you have to haul a wire from the control point back to a panel for each device, why not just use a hard-wire solution like the electro-mechanical systems of days gone by.

The electrical trades – not IT technicians – usually do the wiring. They do not necessarily have the tools or expertise to wire Ethernet. For example, Ethernet requires an 8-wire connection as opposed to only 2 wires for modern control networks. The 8 category 5 wires must be placed in the proper order when wiring the Ethernet connection. Because each Ethernet device requires a point-to-point connection to a switch, there are really 16 wires to terminate and test. Control network wiring, in contrast, uses modern 2-wire solutions that are polarity insensitive (e.g., wire order doesn't matter), which greatly reduces installation time and wiring errors. The same 2 wires often carry both power and data comms.

Control network wiring also is typically done with heavier gauge wire that can support common splice techniques using wire nuts to reduce both first install cost as well as costs associated with moves and changes. By contrast, RJ-45 connectors, the most common form of Ethernet connector, may not be well suited for the environments where control systems are installed.

Additionally, RJ-45 connectors will generally need to be installed on site because 10/100 BaseT cables equipped with connectors cannot easily be threaded through flexible metal conduit (e.g. Greenfield). Installing the connector adds cost to the installation: you have to pay for the connector and for the

time to install it. A 2-wire connection connected to a punch-down or screw terminal block is much faster and requires no connector – just stripped wires. Quality control of field terminated RJ-45 connections is an important factor in calculating installed cost.

In retrofit installations, it may be necessary to use existing wire. Existing wire is seldom category 5 and may not be suited for Ethernet communication (there may not even be more than 2 wires available).

Overall, wiring costs are a significant part of the total system installation cost and are often hidden and unaccounted for when discussing Ethernet-based building automation systems. The cost advantage of a two-wire, non-shielded, polarity insensitive, free topology device network has a multiplicative effect in keeping system costs low - cheaper materials (wire, connectors, etcetera), faster/simpler physical installation, and easier, less-costly troubleshooting.

Noise Immunity

The building automation sector is not limited to office buildings. In fact, a large percentage of building automation systems are installed in much noisier environments, such as industrial, assembly, warehousing, shipping, and medical facilities. In such non-office building environments, there are many electrically "noisy" machines, such as motor/generator sets, furnaces, chillers, medical instruments, electrical transfer switches, etcetera. These can create harsh environments where as much as 2 KV of noise may be injected into the network wires. Ethernet was designed for building environments that are relatively benign compared to environments where control systems are often installed.

Ethernet wiring will "fit" best in control closets and centrally located panels away from harsh environments and the constraints imposed by the wire used for previous equipment. In these locations, RJ-45 connectors will work well, and noise will be very low or non-existent. Additionally, since even in retrofit installations most panel wiring is replaced, the installer is free to use the correct category 5 wire. Some powered Ethernet variants also look promising for future installations.

Amortizing the Costs of the Switch

Ethernet is point-to-point wiring – each device is connected to a switch or hub. While inexpensive switches from companies like LinkSys, D-Link, and Netgear are widely available for home use in the $100 range, these types of switches are not suitable for commercial or industrial control systems. "Hardened" higher-end switches such as Cisco's Catalyst 2950 series would more likely be used. Online prices from CDW for Cisco's Catalyst 2950 switches as of this writing range from ~$700.00 to ~$3100.00 depending on features. Using the most inexpensive example, the switch adds about $30 in part costs to each device. ($700.00 / 24 ports = $29.17 per port) Actual costs may be slightly higher since this assumes full port utilization. A control system with 30 devices would require two 24-port switches, leaving the second switch with 18 ports unused.

The switch also requires a mounting location and power. This may sound trivial and obvious but the rack will add costs and somebody has to pay for the power consumed by the switch. Depending on the installation scenario, it may be possible to use an existing rack and power supplied by the IT department.

Space allocated to data and telecomm infrastructure has increased over time reducing the amount of usable (rentable) office space in a building. Control devices built on an Ethernet communication structure exacerbate the problem. These incremental costs can be calculated on a city-by-city basis based upon the average rent per square foot for your city.

MANAGEMENT

Introducing a Single Point of Failure

Reliability is a key component of any commercial automation system – it is in fact a key differentiator between data networks and automation networks. While most of us would consider it an inconvenience to lose Web access in our office for an hour, none of us would line up to spend an hour trapped in an elevator.

While the reliability of switches is quite high, failure means the loss of communication to all attached devices and the necessary replacement of the entire switch. In other words, the switch is a single point of failure. If a switch loses power, gets hacked, is mis-configured, or simply fails, the control system will fail as well. Unlike peer-to-peer implementations where the failure of a single device causes only a small degradation in performance, a centralized switch failure will cause a complete functional outage for the entire segment of the automation network tied to the failed switch and a correspondingly larger degradation in system operation. While this problem can be ameliorated using redundant switches, doing so adds significant costs to the system.

Who Owns the Switch

It is rare that a single vendor will provide every piece of control equipment in a building. More often than not, the HVAC vendor will be different from the lighting vendor, who is different from the elevator vendor, etcetera. Service contracts tend to be area specific, as well. For example, it is uncommon to find a single vendor who provides both elevator and HVAC service, which leads us to the question: who owns the switch? Who pays for it initially and who pays to repair/replace it? Since company A generally does not want responsibility or liability for maintaining elements used by company B, the situation is usually resolved by each company providing their own switch. This is unfortunate because it means that the spare ports on the elevator switch go unused while the HVAC company may need to install a partially utilized switch to link each end device in their system.

In the end, companies providing sub-systems (even those including devices from multiple manufacturers) would need to include switches in their systems in order to provide adequate quality of service levels.

Sharing and Maintaining the Blended IP Network

Being able to use the existing infrastructure installed for data is often cited as one reason why all devices should simply use Ethernet. While there are certainly tremendous advantages in linking building automation systems with data networks, it is not clear that sharing the data network at all levels in the automation system is a good idea. Data networks are designed to support humans – not machines. The nature of the data network changes daily as computers are added and removed, viruses are injected, and infrastructure components are upgraded or replaced. (How often have you received a message that the firewall will be rebooted at 5:00PM, or that the email server will be taken offline in 10 minutes?) Humans can tolerate such outages, but not a control system.

The Ethernet network used by the control system must be extremely stable; therefore, the blended IP infrastructure of data and automation networks must be equally stable.

Humans have the ability to wait for information. If a web page takes 8 seconds to load because another user on the data network is streaming Internet video, a human is annoyed for 8 seconds, but understands what is happening and will wait for the page to load. If an elevator has to wait 8 seconds for the car position to arrive at the controller while the car is in flight, the elevator car may well crash into the roof or pit of the hoist way. Again, stability and predictable performance is critical for a control system network.

In contrast, a typical device network has reliable data throughput on the order of 15ms or less.

In addition to unknown data bursts, sharing the data network may also be risky due to standard service routines on the data network. It is not uncommon to disconnect part of the network when troubleshooting or doing standard service. Humans usually tolerate such network outages because they were warned ahead of time through some kind of announcement that service would be disrupted, however this doesn't apply to control systems. The last thing you want is an unsuspecting IT technician to power down or disconnect the control system switch while tracking down why the CEO can't print!

IT infrastructure is NOT free and generally NOT suitable for control and automation use. Each port has a hardware and installation cost that must be attributed to a budget in an organization. New operating procedures must also be created to account for the control network's needs. For example, when recovering from a blackout, if power to the switches associated with the operation of the temperature control equipment is not restored first, the servers in the computer room will not be supplied with the 68 degree F air temperature they require to operate reliably.

In the end, a blended IP network will likely require new redundant wiring, switching, and routing – adding significant costs to the entire system.

Assigning IP Addresses

How do you assign an IP address to an occupancy sensor? It doesn't have a keyboard, or display of any kind! A serial or USB port could be provided for configuration, but again, this adds cost to a very cost sensitive device. Adding a serial port requires, at minimum, a connector and a UART. It may also affect the device packaging.

DHCP is an obvious answer, but how do you deal with a device that gets a new IP address when its DHCP lease expires? A building may have 100 occupancy sensors used for security, HVAC, and lighting. It is critical that the controller be able to map the physical location of the sensor to the source IP address of the message it just received. Otherwise, how does the controller know which door to unlock, room to heat, or light to turn on?

One possibility is to give each device a unique hostname and use a DHCP server linked to the DNS server such that each device could always be referenced by DNS name. For example, occ3101.mybuilding.com might represent the occupancy sensor in office 3101. But, how do you get the DNS name into the device? An occupancy sensor has no keyboard, or display of any kind!

Using the Ethernet port is an obvious choice, and is the way many Ethernet devices (most notably consumer grade DSL/Cable routers or wireless access points) provide a human interface for setup. Generally, the device is shipped with a default factory-installed IP address such as 192.168.1.100. The user connects a PC and points a Web browser to the default address to receive a configuration web page.

This works well for single device/single PC installations, but is more difficult to manage when installing 100 occupancy sensors. The occupancy sensors are likely installed by the electrical trade but configured by a technician. If all 100 sensors have the factory default address of 192.168.1.100, how do you know which one is serving up the Web page when you go to configure? If you are adding control devices to the corporate data network, you must take care that a pre-configured device does not conflict with an existing device on the data network that received its IP address through legitimate means. If the company email server, for example, has 192.168.1.100 as its IP address, you wouldn't want to add an occupancy sensor with that same address to the network let alone 100 conflicting IP addresses. This assumes you get a response at all. It is more likely that your browser will see a time-out error because of the conflicting addresses, in which case as many as 99 of the occupancy sensors would need to be disconnected from the network or powered down for the remaining sensor to reply to the browser.

The net result would require a purely serial installation scenario – taking us back a decade before the introduction of ANSI/EIA-709.1 and control network databases – effectively causing installation costs to skyrocket.

UPnP

Assuming devices have the resources to support TCP/IP and a minimal Web server, and assuming a device can establish an IP address through some automated fashion, UPnP announcement and discovery protocols such as GENA and SSDP running over HTTPU or HTTPMU look more promising, but are only partial solutions. For instance, mapping "found" devices to specific hardware is not supported in existing UPnP announcement and discovery protocols.

UPnP is great for finding devices on my home network; and because I only have one PVR, one access point, and named PCs on my network, the list of "found devices" is obvious to me. But what if the list included 100 occupancy sensors? How would I know which one is which?

The problem of address assignment has been solved already (with varying degrees of elegance) by a number of control networking systems. ANSI/EIA709.1 networks, for instance, even provide built-in mechanisms that help standardize the process.

However, the problem remains unresolved in an IP-based automation network and must be considered in the context of an overall solution. The address assignment mechanism must be common for each device, and if the control system is to include devices from multiple vendors, the address assignment mechanism must be agreed upon by each of the vendors as well. Note also that if the solution involves the use of DHCP or DNS servers, the cost of those servers has to be amortized over the number of nodes in the system, as does the ongoing support of those servers.

Request/Response

Web services implemented using SOAP are built on a request/response paradigm. This makes perfect sense when you consider their origin. Web services were created in response to the need to let one machine "talk" to another machine in a way that could be supported on the same infrastructure used for serving Web pages. A machine consuming a Web service exposed by another machine is directly analogous to you pointing your browser to www.yahoo.com and reading the Web page served to you by Yahoo's server.

You probably used Internet Explorer running on Windows to get the Yahoo Web page. Yahoo uses FreeBSD servers to provide the page. The common denominator between Windows and FreeBSD (and every other computer out there) is ASCII text. Your browser makes a text request, and Yahoo responds with a text page. The text contains embedded metadata (HTML tags) that instructs your browser how to render the page so you can read it.

Similarly, a computer that consumes a web service makes a text request to a server, and the server responds with text. Only, instead of the response being HTML, it is XML.

So how does this play out with occupancy sensors? Does the lighting controller, HVAC controller, or security system have to constantly poll all 100 occupancy sensors to know if there is motion in the building? Today, unfortunately, the answer is yes. And to make matters worse, depending on the web service implementation in the device, the time required to construct and tear down SOAP transactions can be significant.

Various schemes to overcome this limitation have been proposed, but each use some kind of poll request. Web services simply do not provide an "eventing" mechanism such as those typically found in advanced control networking protocols – which makes peer-to-peer communications using web services very difficult to implement.

WS-Eventing

The WS-Eventing working group is currently addressing this limitation for business systems, but it is not clear that the solution will map well to the needs of control systems. WS-Eventing would essentially allow the controller to register with the occupancy sensor as an event sink such that the occupancy sensor could call an exposed Web service on the controller each time it notices movement in the room. This is obviously better than having the controller poll each occupancy sensor, but still requires the occupancy sensor to have the Web services stack and event updates are still limited by the amount of time it takes to establish and tear down SOAP transactions. In this type of network architecture, the controller serves as a big Web server. Each device sending information is analogous to a browser request, only instead of returning a Web page, the controller returns an acknowledgement that it has received the WS-Event.

"Chunky" Vs. "Chatty"

A "chatty" system is one that fetches a single piece of information at a time. Imagine populating a spreadsheet by fetching data from a database one record at a time. Get record #1, get record #2, get record #3, etcetera. The spreadsheet would eventually populate, but the network traffic between the spreadsheet and the database server would be enormous. By contrast a "chunky" approach would fetch all [or many] records in the database in one transaction, and let the spreadsheet application

decide which and how to display the records for the user. The advantage of a "chunky" call is that it avoids the overhead of constantly setting up and tearing down transactions between the spreadsheet and the database server – many records are received in a single transaction.

Web Services are designed to use "chunky" calls, but this does not map well to control system messaging. When the occupancy sensor sends information to the controller, there is only a small amount of information. Any single occupancy sensor is not "chatty" per se, because it does not often need to send information, but 100 occupancy sensors sending information to a controller has the effect of a "chatty" conversation from the controller's perspective. This leads to higher system costs as controllers will need to be powerful enough to establish and tear down many transactions per second to keep up with the flow of information from all the devices in a system.

SECURITY

Depending on the installation scenario, it may be necessary to include a control system firewall alongside the switch. The cost of the firewall needs to be amortized across the number of nodes in the system to determine the incremental cost per node. Keep in mind that firewalls need an administrator, so those costs should also be considered as part of the overall installed system cost.

Related to adding firewalls to the system is the issue of peer-to-peer security. Depending on the type of control system and/or site, encrypted or authenticated transactions between devices may be required. IPSec or HTTPS, commonly used in IP networks, may suffice, but again add cost to the overall system. Both IPSec and HTTPS place additional computing burden on the device and require extended memory.

An intangible element that contributes to system security is the number of hackers familiar with the underlying protocols that drive the system. TCP/IP is the most widely used protocol on the planet, and thus has the most people with some knowledge of how it works and how to exploit its weaknesses. There are far fewer people familiar with control system protocols and thus a smaller pool of potential hackers. Imagine the power of the siren's song to hackers created by the opportunity to have a high-rise office building's lighting system spell out political or social messages during the night. Governments, including the US government, worry about the far more sinister effects of hacking into control systems, particularly at facilities that house many people.

Opportunity

By contrast, ANSI/EIA709.1 networks take advantage of the authentication capabilities built into the base microcontroller; which, while not as secure as IPSec or HTTPS, provides adequate protection at a reasonable cost point since the network itself is isolated from the business' IP network except at the gateway/supervisor which in turn is hidden by the corporate firewall. The net effect is a three-tiered "perimeter": i) corporate firewall, ii) security of the gateway/supervisor, and iii) embedded authentication of the device network protocol.

XML/Web Services at the Gateway/Supervisor Level

At the gateway/supervisor level, the challenges described above are largely overcome using modern supervisors, or gateways, built around powerful 32-bit microprocessors that have sufficient amounts of flash and RAM to host a TCP/IP stack, HTTP, SOAP, an XML Parser, etcetera. The gateway/supervisor acts as a proxy, or bridge, between the control network and the web services world. The net effect to the enterprise is that control network devices may be monitored/controlled using web services through the gateway/supervisor – transparently. Acting as a proxy, the gateway/supervisor may also implement access strategies preventing misuse of the control system.

In addition, many modern 32-bit microprocessors have built-in support for serial and USB, making it easy to support various setup tools without adding extra cost to the gateway/supervisor.

A system that contains relatively few gateways/supervisors thus eliminates the need for a dedicated set of Ethernet switches. Assigning addresses can be done in a way similar to setting up a PC, since a screen and keyboard can be plugged into the gateway/supervisor's USB or serial port.

Furthermore, because such gateways and supervisors limit access to the underlying automation network, they act as an additional layer of defense against hackers.

Using XML and web services at the gateway/supervisor level provides integration benefits when dealing with mixed systems consisting of open devices networks, legacy sub-systems, and proprietary closed sub-systems. In fact, such a layered architecture perfectly displaces some of the current standards for building automation that reside in the gateway/supervisor – and does so more simply, cost effectively, and openly.

If the underlying field devices use a standard peer-to-peer control networking protocol such as ANSI/EIA 709.1, the loss of the gateway/supervisor or the Ethernet switch does not halt the control system. In addition, as the system changes, a common field bus that can truly support ad-hock network architectures where additional bus segments and devices may be added anywhere along the wire provides a huge benefit over the life of the system.

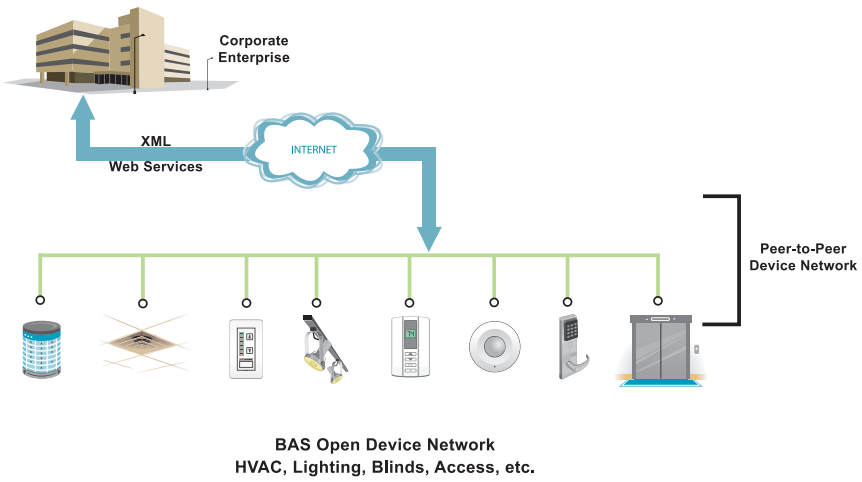

Figure 2 – Architecture for Maximum Benefit Combining Open, Interoperable Device Networks and XML/Web Services

The ultimate opportunity for delivering value and performance utilizes an XML/Web Services layer on top of truly open, interoperable device networking system as shown in Figure 2. Such a system provides:

❍ Simpler integration when creating a complete building automation system consisting of:
- Legacy systems;
- Closed/proprietary systems; and
- Stand-alone devices.

❍ Necessary segregation of the IP and automation networks resulting in:
- Higher reliability of both networks;
- Greater quality of service;
- Lower business costs;
- Lower infrastructure costs and complexity; and
- Lower first-time and life-cycle costs of the automation systems.

❍ Benefits of open systems in the area that has hurt end-users (building owners) for decades, such as:
- Freedom to choose amongst best of breed automation suppliers;
- Lower installed cost and life-cycle costs for the building;
- Faster, cheaper, more reliable integration; and
- Open bidding/procurement for system extension, maintenance, and upgrades.

XML web services can deliver integration but not interoperability. The distinction is the difference between getting the benefits noted in the above list and getting sold a Trojan horse. Whether vendors of closed proprietary systems choose to embrace true openness and distinguish their products on price, feature set, quality, etcetera, or choose to use XML and Web Services as a Trojan horse to "claim" to be open is yet to be seen.

Conclusion

Disruptive technologies like XML and Web Services are true change agents for industries and can appear seemingly overnight. In the buildings industry, moving from proprietary systems to an open Web Service based approach will be a huge step forward. Devices located at the gateway/supervisor layer have the necessary hardware capabilities to support XML web services and are the natural integration point among disparate automation sub-systems.

Yet, XML and Web Services by themselves cannot deliver the full range of benefits provided by today's open device networking technologies, nor do they deliver on the benefits that many would say they do when used at the device level. However, when used in conjunction with device networking technologies such as ANSI/EIA 709.1, XML and Web Services deliver real economic benefits to all members of the building automation industry.

Applied correctly, as a layer on top of truly open, interoperable device networking systems, XML and Web Services are indeed good for everyone and will ultimately allow end-users, device and system manufacturers, and integrators to reap the full benefits of IT technologies in the automation world.

550 Meridian Ave.
San Jose, CA 95126
tel: +1 408 938 5200
fax: +1 408 790 3800

www.echelon.com