Home

Web Services For Building Controls: At A Crossroads

By Mr. Toby Considine
May 4, 2005

Two years ago, controls industry leaders came together at BuilConn to launch oBIX, an initiative to bring control systems to the enterprise by leveraging the best practices of the data center. In the IT world, innovators were no longer building bigger systems, but were instead developing approaches to orchestrate many, many, independent systems. Monolithic, single purpose systems tend to be inflexible and brittle in the face of change. Systems built by dynamically federating components from independent systems remain nimbler than the monolithic systems, better able to adapt to changing requirements, and operate more effectively across corporate boundaries.

This revolution was driven in part by near-ubiquitous adoption of XML. Because XML is self-describing and readable, interface problems could be resolved quickly. This revolution had many names, from service oriented architecture (SOA) to grid computing. A key outgrowth of XML adoption is the widespread adoption of Web services, using XML to send messages between systems. The most broadly adopted protocol for XML messages is SOAP. Another key component of this revolution was directory enabled networking, wherein organizationally aware security permeated the fabric of systems. As these security systems spanned organizations, it grew into Federated Identity Management, wherein a single user's identity spanned the system and organizational boundaries.

In the post September 11, 2001 world, government embraced these approaches as well, as new organizational structures and new challenges demanded that agencies be able to share information and integrate processes across traditional boundaries. Large, complex processes, encoded in monolithic systems internal to each agency had to be re-crafted to share information. This new openness did not mean, however, that these agencies could relinquish their business processes, often legally mandated.

Building control systems, however, remained mired in domain-specific protocols. These systems could participate with other systems only in limited ways, and with great effort. Even the wire protocols did not play well in the new world where the isolated control network was a thing of the past. The simplistic security of the control system, once stripped of isolation, provided almost no security at all.

XML and SOAP, together with other the domain-specific services specifications, define an open standards-based approach able to leverage the world of open standards and corporate security to integrate disparate systems. Adoption of Web services is a clear opportunity for the building controls world to enable integration with enterprise systems, making control operations and information

visible to the enterprise. Control systems must become first-class citizens on the corporate "enterprise service bus" by embracing the session-less protocols of the Web.

At BuilConn this year, we saw that the first step is near fruition. Just about any kind of control system can produce and interact with Web services. oBIX demonstrated interoperability and described the path to the first release of its standard. BACnet demonstrated a capability and openness in its Web services that almost no one would have anticipated two years ago. Building systems integrators are gaining experience in setting up various LON-based Web services. Nearly every control system has some sort of XML interface. Walking around the trade show, one can find numerous Web service gateways to get to the few systems without Web services.

With all this progress in Web services, control systems are still not ready for the enterprise. They remain controls-centric. They do not interact well with other enterprise systems, or even with each other. Great progress has been made, but there remains much more to accomplish.

We have had many conversations about this recently, both within oBIX and with other interested parties. At the risk of being unfair to some, I will try to simplify and condense the three perspectives: those of controls professionals, of IT professionals, and of building operators. While the views below are extracted from actual conversations, please realize that the originals were longer and more cogent; the oversimplifications and mistakes are probably mine.

Controls Standard Developer Perspective

We are using Web services to model control interfaces against which applications can do fine-grained gets/sets. A single schema works for all variety of devices. We also define a registry to allow for discovery of data by traversing a tree-structured hierarchy of controls data. This hierarchical abstraction allows access to all of the primitive sensors and actuators in a building/campus. This Web service is the endpoint for the IT skill set, there is nothing behind it the IT world can talk to.

The last thing we want is a schema for each kind of device out there. We want one schema that could model them all using a small set of primitive values (because control is that simple). It is a very important goal of oBIX to have a normalized view of the entire controls world. We have no standard lexicon and won't for some time.

There are thousands of devices with different configurations; very few HVAC devices have the same configuration. The device manufacturers and vendors, if they even still exist, are not going to go back and create device profiles for products they no longer sell. The IT community won't be creating profiles for these devices because they can't even talk to them. The burden will fall upon gateway servers, and the issue is, how easy will it be for these servers to act as proxies for all of the different devices behind them?

Because of the simplicity and elegance of our interface, we will be able to auto-generate a good deal of the Web service interface, and move quickly to deploy our Web services everywhere. It is important that we do not limit what our Web services can be used for. In the future, controls consultants will be able to get to control systems securely without visiting the customer site. Any restriction on the granularity of access will prevent us from getting the full benefit of Web services.

IT Professional / Systems Architecture Perspective

Note: In general, anything that begins WS is a Web services specification. WSRF is the Web services resource framework; WSDM is Web services distributed management; CIM is the common information model (for computer/network resources); and DMTF is the distributed management task force.

Web service is all about trying to figure out what "minimal" model is needed to allow client applications to interoperate with services with minimal shared understanding. The more shared understanding that is required, the more brittle the system, and the higher the bar is to interoperability.

The fundamental question when using Web services to model an application domain is "What abstraction are we presenting in the application programming model?" A simple hierarchy of values requiring a standardized path traversal agreement requires the enterprise programmer to understand control systems. This model makes it more difficult for any individual to effectively determine how to integrate enterprise systems with building controls systems (the number of people who understand both domains is very small). The preferred model is an abstraction layer on top of physical devices that presents the "capabilities" of the system, encapsulating the details of the physical devices. The abstraction layer pushes down onto the aggregation of the lower level physical sensors and actuators.

We have a similar situation when managing devices in a data center. The goal is to find a common interface to data center components that is sufficiently interesting to motivate third-party application software vendors to address. There is significant variation in servers, printers, network controllers, etc. However, there is a common core data model that provides some level of interoperability/uniformity. This core data model is extensible, allowing vendors to provide specific operations and properties if required. With a standards-based approach, third-party application providers feel confident to provide value added functionality.

This is similar to the problem of building controls. A domain (e.g., an office, a floor of an office building, an entire building, an entire campus) has certain "capabilities," often described as a collection of resources. When each resource has a well-defined XML schema, one is able to model how an application interacts with each resource. A Web services interface provides me with further information about the set of operations an application can invoke on that resource.

In a data center, resources have a life: they are created, deployed, brought on/off line, crash, are un-deployed, and finally destroyed. It appears that a similar lifecycle exists with building resources (however, perhaps at a timescale that is "longer than data center resources"). Different domains would have registries (and potentially naming services) indicating which resources are available in that domain. For example, an office has a light resource, a HVAC vent resource, a temperature sensor resource, a thermostat resource, etc. Applications would interact with resources, and "discover" them within a domain.

In WSRF (and therefore WSDM, Grid, other exploiters) we have chosen to model the set of related data items and their values as an XML document associated with a particular resource. So some resources are simple (one or two data values). Some resources (like a data server) may have hundreds of properties. In either case, that "bag of state," modeled as a single XML instance document, and validateable to an XML schema type associated with the "resource type," is the thing of interest to the system. The individual resources are important to consider, separately from the notion of how the resource relates to other resources (either physically or logically grouped together). There are separate techniques available for applications to reason about the relationships between the resources.

WSRF is just a Web services framework; it is completely neutral to how it is composed with the design of the specific resource and its XML schema. WSDM is the same. CIM (DMTF) is the source we look to for a definition of the XML schema properties for the actual "concrete" resources like servers, operating systems, etc.

Building Operator Perspective

Do Web services define interactions all the way down into the control world by developing definitions to cover every controller and control process that might be implemented? If so, then it's no wonder the controls industry is taking so long to implement a WS environment. All of our discussions have been about using existing building controls to handle control-level transactions using LonWorks or BACNnet (or something proprietary or legacy), but to provide access to those transactions at an enterprise level by turning the existing "control object" into a fully encapsulated "service."

With my not being an IT guy, the discussion is a little hard to follow. I do think that the concept of defining the commonality instead of the myriad details does make any "conversion" or "porting" of control system transactions (PID loops, optimal start, local demand management, or even sensor calibration or historical data collection, or many others) a much more manageable task. If you try to define every detail of each control system as a Web service, then not only every vendor but perhaps even every enterprise may end up having to develop a custom WSDL. If oBIX can provide a layer of abstracted commonality that is applicable for all vendors and then provide a detailed guide for vendor specific extensions that allow vendors to add value to their own systems, while at the same time providing the framework to define how legacy systems need to bundle their existing data and transactions into an open WS environment, then that would meet most of my goals.

I [the building operator] am in favor of abstraction at the enterprise level, even if the devil remains in the details at the building control level. I would like to be able to tunnel to the details from a remote location, but that may be an appropriate use of vendor-specific tools in any case and not an OBIX thing at all (except maybe defining what a tunnel consists of).

My Perspective

There is still a lot of distance between the three perspectives above. We need further work within oBIX to bring them together. We now have at least three WS-building controls standards that are either useable or will be within a year. LON, BACnet, and oBIX can all bring control systems to Web services. None of them is, yet, an enterprise interface to controls. Between these three, we have WS-building controls, a complete definition of what the control system engineer needs from Web services.

Where do we go from here?

WS-building controls, as defined above, is not enough.

I still want the abstraction that the enterprise needs. Few enterprise programmers are trained in controls; nor do they want to be. The work has been important, but it is not enough. We simply do not have the taxonomies in place to make things useful to the enterprise. We also need to expose these abstractions in a manner that can exploit other Web services standards, such as WSRF.

One critical piece may be coming already. ASHRAE Guideline Project Committee 20 (GPC 20) will provide the HVACR industry with XML definitions. Standard definitions will be freely available

from the ASHRAE website. Publication of Guideline 20 is expected by 2007. If we accept GPC 20, then we have the lexicon we need for one domain. We need to identify other bodies to define the lexicons for the other building controls domains.

I propose we define oBIX v2 as the application of common abstract lexicons to each domain under oBIX. The focus of oBIX v2 will be to provide to the enterprise the specific business needs of scheduling, M&V, performance, commissioning, etc. A profile of the oBIX v2 framework, then, might look like:

Foundational protocols, used by controls professionals and systems integrators

- WS-buildings controls:
- BACnet-WS;
- LonWorks-WS;
- oBIX v1; and
- Proprietary Web services.

Functional domain services used by the enterprise: owners, operators, and tenants

- WS-buildings HVAC (based on GPC 20);
- WS-buildings power;
- WS-buildings access control;
- WS-buildings intrusion detection;
- WS-buildings CCTV; and
- WS-buildings occupancy.

Higher end integrated applications interacting with the domain services and their exposing functionality through Web services

- WS-buildings performance (M&V, commissioning);
- WS-buildings analytics; and
- WS-buildings tenant services.

oBIX v2 should also compose with Web services specifications for policy, security, etc., in a fashion similar to that used by WSDM, WSM, et al. oBIX v2 should also include standard guidance for control silos coming to oBIX, including:

- Standards for other industries to implement their own WS-buildings services;
- Reference implementations of oBIX as WSDM and WS-management; and
- Reference implementation for a combined oBIX/UPnP service.

To accomplish this, oBIX will need wider participation. Enterprises outside the controls manufacturing world with interests in buildings and their operation need to participate. We need more participants with enterprise experience to join the oBIX committee. We need more business managers to contribute use cases to make sure we get it right.

Interested parties can learn more about oBIX at www.oasis-open.org. **IBT**

Considine is co-chair of the OASIS oBIX Technical Committee and has been working with enterprise applications and integration of embedded control systems for 20 years. As a systems specialist in facilities services at the University of North Carolina, he has struggled with the network demands,

poor data sharing, and non-scalable security issues posed by last-generation control systems. This led to his focus on open discoverable data standards for control systems. Before coming to UNC, he worked to integrate other silo processes into the enterprise for companies including The Architect's Collaborative, Reebok, and Digital Equipment Corporation.