Subsystems Integration

1. Introduction

The increasing technological sophistication of building systems means they can affect a business's bottom line - for better or worse. Creating a high performance building - an effective building that operates efficiently, and attracts and retains occupants with its amenities and benefits - is something owners want, and it can be achieved by integration. But effective integration is not just a matter of tying subsystems together.

The idea of different systems with different responsibilities working together as if part of a team - addresses the fundamental aspect of having an effective building. It involves applying technology to unseen portions of a facility in the interest of creating a better environment for those working there, and better processes. When done properly, successful integration shows up on the bottom line of financial reports by reducing the acquisition cost and the life cycle cost of facilities.

The integration of building systems is not a new concept. However, philosophies of integration vary. Not too long ago, it was a feat for more than one building system to be connected to relays, allowing on/off scheduling through the installation of a time clock. Integration took on a new meaning in the late '70s when computers became integral parts of businesses, and building systems, adding new dimensions to facility management. In today's market, integration is not only seen as a tool that can improve the overall performance of the facility, but also improve the performance of the people and processes that affect an organization's livelihood. An enterprise-wide approach to integration can turn something we weren't too concerned with takes on new implications. For that reason, interest in integration today is stronger than ever.

The most common subsystems that can be integrated are the automation and control subsystems, which manage specific domains that contain a number of components that regulate the overall environment including HVAC, electrical, energy, fire, and security. Traditionally this was done strictly from a supervisory perspective. The advent of the microprocessor has brought more sophistication to these components, along with the ability to communicate electronically. This communication leads to direct connection of the system via a data port, as opposed to the traditional supervisory approach via relays and sensors. Evolving standards in the world of data systems bring another dimension to integration: integration of the building systems to the data system. The most frequent occurrence involves the use of a common data infrastructure for all systems in the enterprise. Whether it is the business information system, the voice system, the video system, or the BAS, each resides on a common, integrated information infrastructure. An important integration occurs when data from

the building systems is integrated with data from the business process itself.

This interest in the integration of distinct building systems into a common, cohesive operating unit derives from a number of different perspectives. From the daily operating perspective, integrating building systems and data systems will enable better management of information that can impact the facility performance, for example, the building systems are there to serve occupants and they don't need to operate when no one is in the building, by integrating various subsystems with a BAS, building occupancy can be determined, for example, by a card access system or by occupancy sensors in the occupied areas. From the whole enterprise perspective, on a larger scale, integrating building systems with data systems bring another source of functional integration: merging important information from the building systems with information critical to the operation of the business. Consider the example of a pacemaker manufacturing facility. In this facility, all data pertaining to the manufacture of the product is retained for historical purposes. If the details of a specific pacemaker's manufacturing process needs to be reviewed at some point in the future, the more data retained from the past, the better. To satisfy this requirement, a tremendous quantity of historical data gathered by the building systems is merged with information from the plant management system. A review of this information would show:

- Source of raw materials – *from plant management system*
- Source of supplied components - *from plant management system*
- Who did the assembly - *from plant management system*
- Who did quality assurance inspections - *from plant management system*
- Was the environment within tolerance - *from building system*
- Were there any building alarm system abnormalities – *from building system*

The integration of information from different systems into a common database is instrumental to the mission of the business in this interactive world of business.

2. Field and Controller Levels Integration

In this highly sophisticated world of intelligent automation and various control systems, broad knowledge and expertise is continuously being needed to create products that build on the concept of the intelligent environment and the realization of smart control applied to various industries.

Although incorporating microprocessor-based controllers/actuators/sensors in these products to achieve highly distributed control, in the beginning these products were

designed to be manufacturers specific. We saw that with the increase in popularity of networked automation and energy management systems, communications ports were added to these products and various communications protocols evolved, and we have studied several of them so far.

The earliest of these protocols were again, often proprietary to the equipment manufacturers. Later, as specifications began to require that mechanical and electrical equipment provide serial interfaces, several of the most common protocols became de facto standards. Modicon's MODBUS and OPTO-22's OPTOMUX are two examples. Eventually, several industry standards bodies formed committees to define protocols that would be available to be deployed without licenses or royalties (ASHRAE-BACnet, OPC Foundation-OPC, Profibus International-Profibus, etc.). As object-oriented programming paradigms gained widespread acceptance, it gave birth to the current generation of object-oriented protocols - BACnet, LonMark Functional Profiles defined for LonWorks, and European Installation Bus Object Interface Specification (EIB - ObIS), among others. Refer to BACnet object example shown in Figure 1, and Figure 2 for LonMark object.

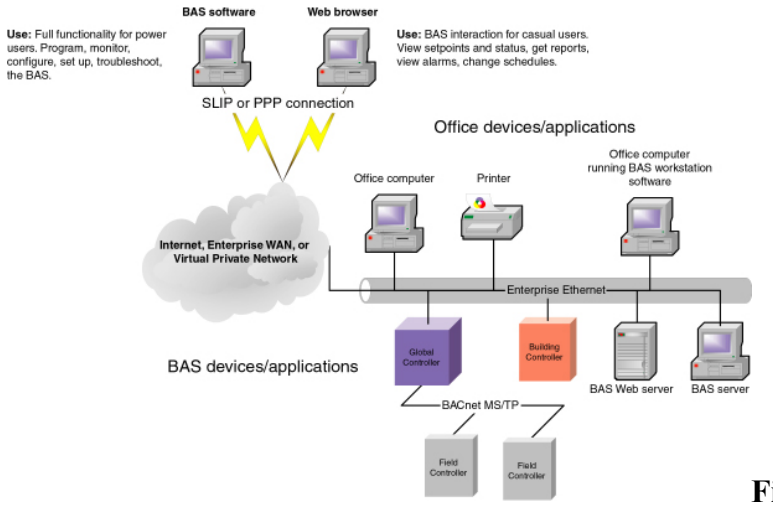


Fig. 1a BACnet network

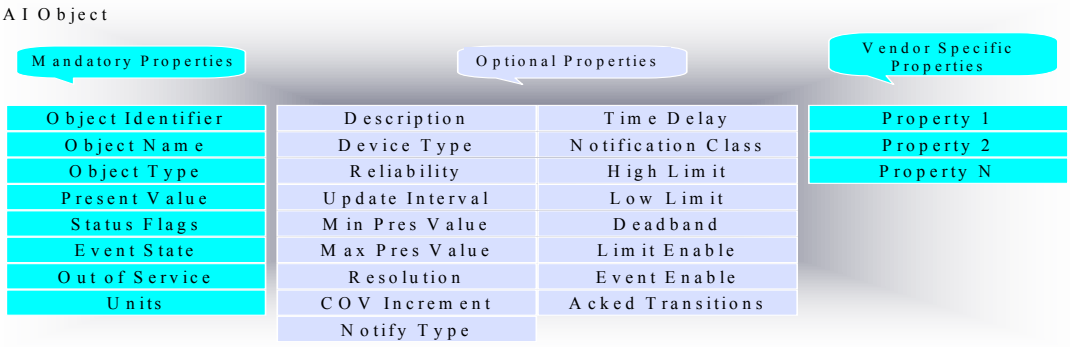
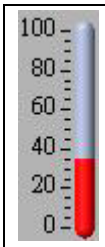


Fig. 1b Object Properties – mandatory, optional & vendor specific



Object_Name	Space Temp
Object_Type	Analog Input
Present_Value	37.5
Unit	Degree Celsius
High_Limit	50
Low_Limit	5

Fig. 1c Example analog object properties

BACnet *objects* are manipulated through Services, among them is *Object Access Services,* which include: *ReadProperty, WriteProperty, CreatObject, DeleteObject, etc.* The following VB code example in Figure 1d illustrates how vendor's supplied BACnet dll can be used together with Excel spreadsheet .

This example demonstrates how to read Analog and Binary values into Microsoft Excel using vendor's BACnet dll. When the worksheet opens it initializes the Application Layer to use BACnet/IP giving the device ID 200 and a network number of 1. It then attempts to read the present value of Analog Values 1 to 100. It then attempts to read from Binary Values 1 to 100. When the excel spreadsheet is open, it acts as a BACnet device on the network with a device ID of 200. Any BACnet device on the network including a BACnet workstation can read/write to the excel spreadsheet. The Excel Spreadsheet can also be set up to read/write from any BACnet device on the network.

When the vendor's dll is available, open up Microsoft Excel and go into the Visual Basic Editor then configure the vendor provided BACnet dll library to be included in the Tools-Reference. Open up the source code window for **ThisWorksheet, then type in the following similar coding example.**

```
Option Explicit
Private objApplicationLayer As New BACNETXLib.ApplicationLayer
Private Sub Workbook_Open()
    Static LoadedFlag As Boolean
    Dim i As Long
    If Not LoadedFlag Then
        LoadedFlag = True
        objApplicationLayer.Initialize "DeviceID=200;IPEnabled=1;IPNetworkNumber=1"
    End If
    Cells(1, 1) = "Instance"
    Cells(1, 2) = "Analog Value"
    Cells(1, 3) = "Binary Value"
    For i = 1 To 100
        Cells(i + 1, 1) = i
        ReadAnalogValue i
        ReadBinaryValue i
    Next
```

```
End Sub
Private Sub ReadAnalogValue(Instance As Long)
    'This example demonstrates the use of the readProperty service
    'by reading the present value property from Analog Value number 1 inside device 200.
    'If the service is successful, then the value of the property will be
    'stored in the ack object.

    Dim Service As New ReadProperty
    Service.async = False
    Service.DeviceID = 200
    Service.Request.objectIdentifier.Instance = Instance
    Service.Request.objectIdentifier.ObjectType = BACnetObjectTypeAnalogValue
    Service.Request.propertyIdentifier = Property_presentValue
    Service.Execute

    If Service.Error.choice = BACnetErrorChoiceEmpty Then
        'No Error - Display the results
        Cells(Instance + 1, 2) = Service.Ack.propertyValue.Real
    Else
        'An Error has occured
        Cells(Instance + 1, 2) = "Error Class = " & Service.Error.Error.Class & " Error Class = " & Service.Error.Error.Code
    End If
End Sub

Private Sub ReadBinaryValue(Instance As Long)
    'This example demonstrates the use of the readProperty service
    'by reading the present value property from Binary Value number 1 inside device 200.
    'If the service is successful, then the value of the property will be
    'stored in the ack object.

    Dim Service As New ReadProperty
    Service.async = False
    Service.DeviceID = 200
    Service.Request.objectIdentifier.Instance = Instance
    Service.Request.objectIdentifier.ObjectType = BACnetObjectTypeBinaryValue
    Service.Request.propertyIdentifier = Property_presentValue
    Service.Execute

    If Service.Error.choice = BACnetErrorChoiceEmpty Then
```

```
            'No Error - Display the results
            If Service.Ack.propertyValue.Enumerated = BACnetBinaryPVActive Then
                Cells(Instance + 1, 3) = "On"
            Else
                Cells(Instance + 1, 3) = "Off"
            End If
        Else
            'An Error has occured
            Cells(Instance + 1, 3) = "Error Class = " & Service.Error.Error.Class & " Error Class = " & Service.Error.Error.Code
        End If
End Sub
```

Fig.1d BACnet Analog Input object, with coding example

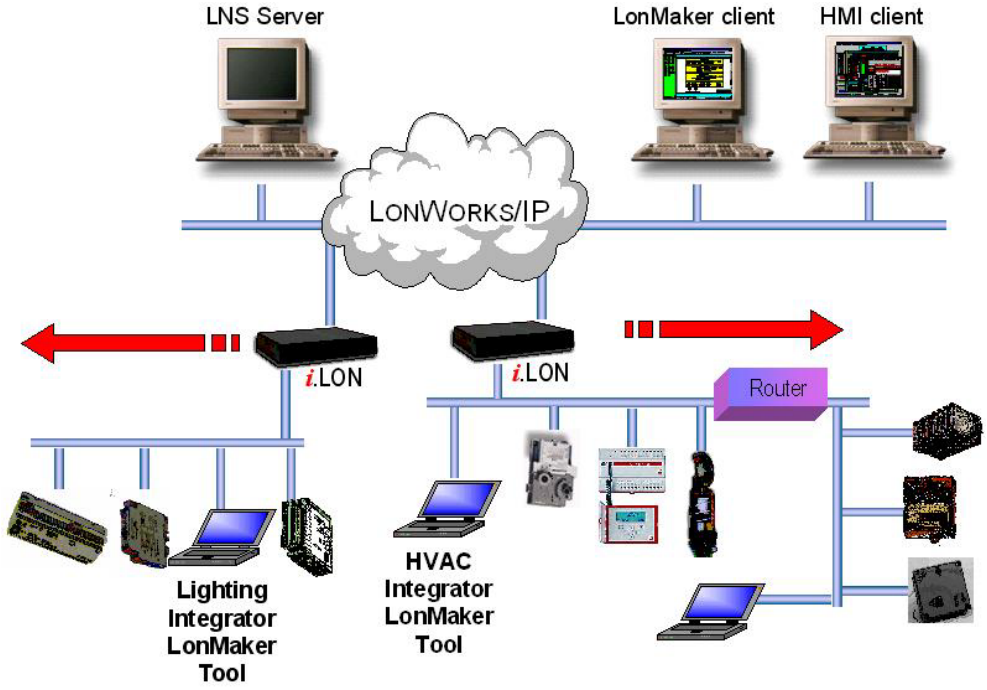
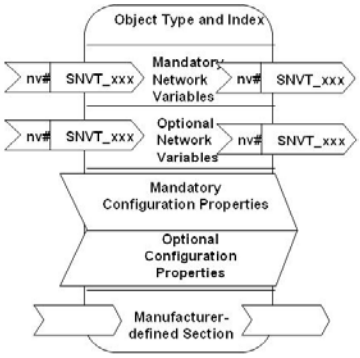
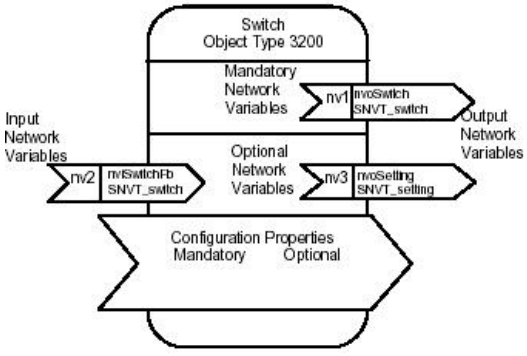


Fig. 2a LonWorks network





```
// Set LonMark Self-Documentation
#pragma set_node_sd_string "@3.0&0,3"
#pragma set_std_prog_id 81:23:45:12:34:12:34:12

// LonMark Node Object Mandatory NVs
network input sd_string("0|1") SNVT_obj_request
network output sd_string("0|2") SNVT_obj_status
      nvo00Status = {1};

// LonMark Open Loop Actuator Mandatory NVs
network input sd_string("1|1") SNVT_switch
      nvi01Value;

// I/O Declarations
IO_0 output bit ioRelayControl;

// Process Node Object Requests
when(nv_update_occurs(nvi00Request))
{
      // reject all LonMark status requests
      nvo00Status.object_id = 1;
      nvo00Status.invalid_request = TRUE;
}

// Process Actuator Object Updates
when(nv_update_occurs(nvi01Value))
{
      if (nvi01Value.state == 0)
            io_out(ioRelayControl, 0);
      else
            io_out(ioRelayControl, 1);
}
```

Fig. 2b LonMark Object with Lamp Actuator Object and example code

To understand where we're headed, it is helpful to realize just how far our industry has advanced. In the 1980's, there wasn't a protocol that could meet the broad needs of various control industries (e.g. building, industrial, transportation, etc.), even for the majority of a Building Automation system's needs. As we press forward into the 21st century, our problem is not one of "not enough" but one of "too much." For example, we now have not one, but several standard protocols that can meet the needs of an average building automation system. The various building automation system manufacturers build products that support one or more of the standard protocols that are available, however, integrating more than one protocol into a single system can be a challenge. Although the data structures for the standard protocols are similar, their implementations are quite different. Gateways between any two of the standard protocols tend to be complex and less flexible, and bridging more than two can become unwieldy, as seen from Fig.3 for a typical building automation system.

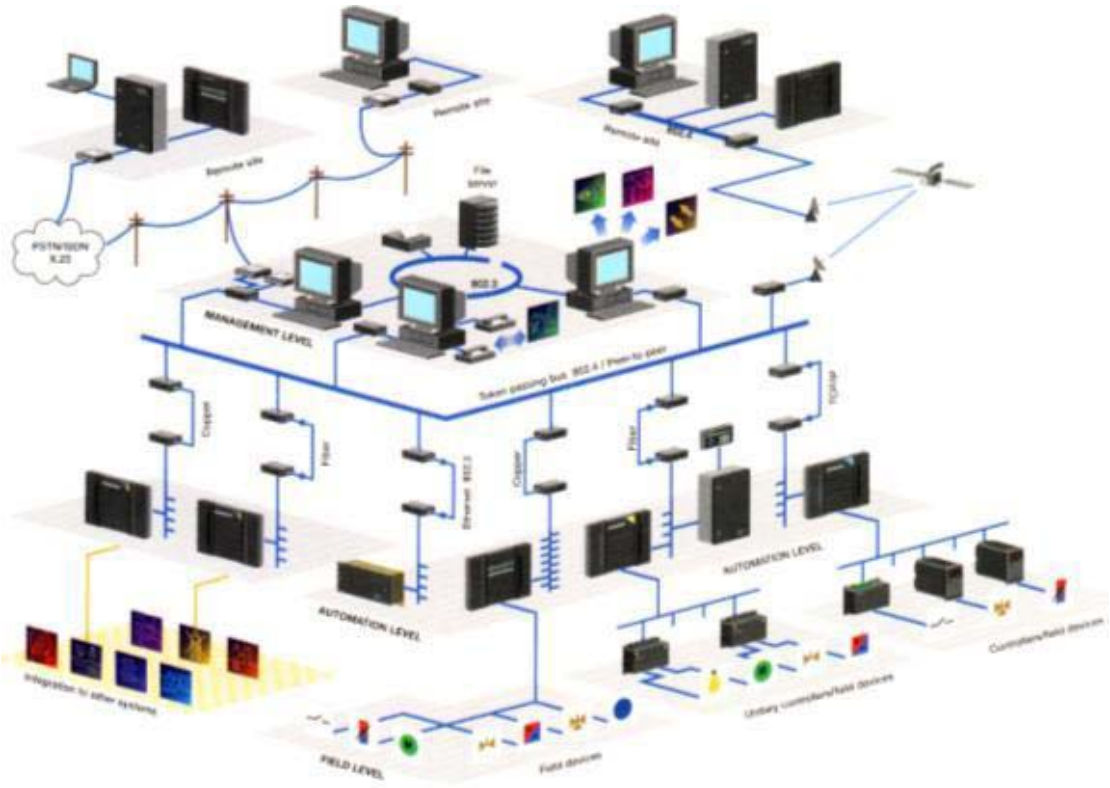


Fig.3 Typical building automation system

A second problem area stems from the fact that automation systems are increasingly seen as a part of a much larger information system, as shown in Fig.4. For instance, facilities managers now routinely have specialized software for managing their tenant spaces, their assets, their equipment maintenance, and even for their energy procurement. The requirements for "information integration" are now much broader than when the standard protocols and their corresponding data models were conceived.

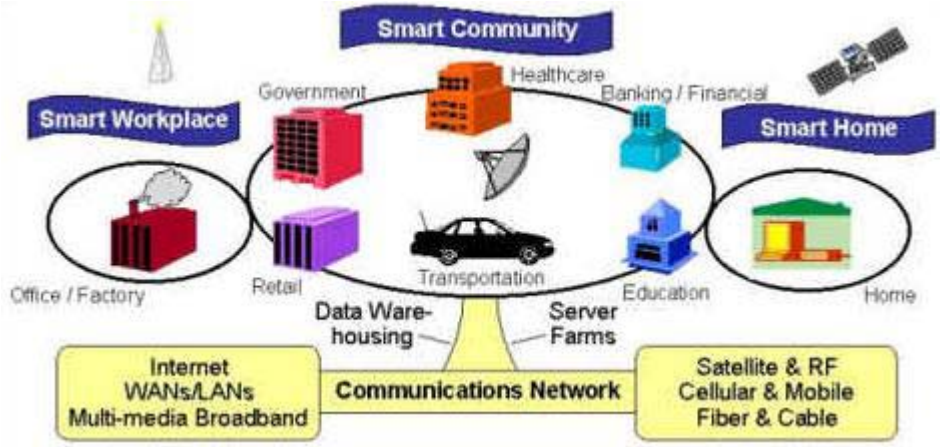


Fig.4 Automation is becoming a vital part in information system

Therefore, if BACnet, EIB objects, and LonMark functional profiles are methods of modeling information, what is needed is a unified information model to include these protocols as well as other automation-related applications. That is, for device-to-device interface, or at a zone or unitary-controller level integration, it is practical to apply these automation standard control protocols to provide the **WHAT** to communicate among different subsystems to make them interoperate (An example is shown in Figure 5 for a LonWorks network), and let the information model to formulate the **HOW**. I refer to the HOW in the next section – Management Level Integration

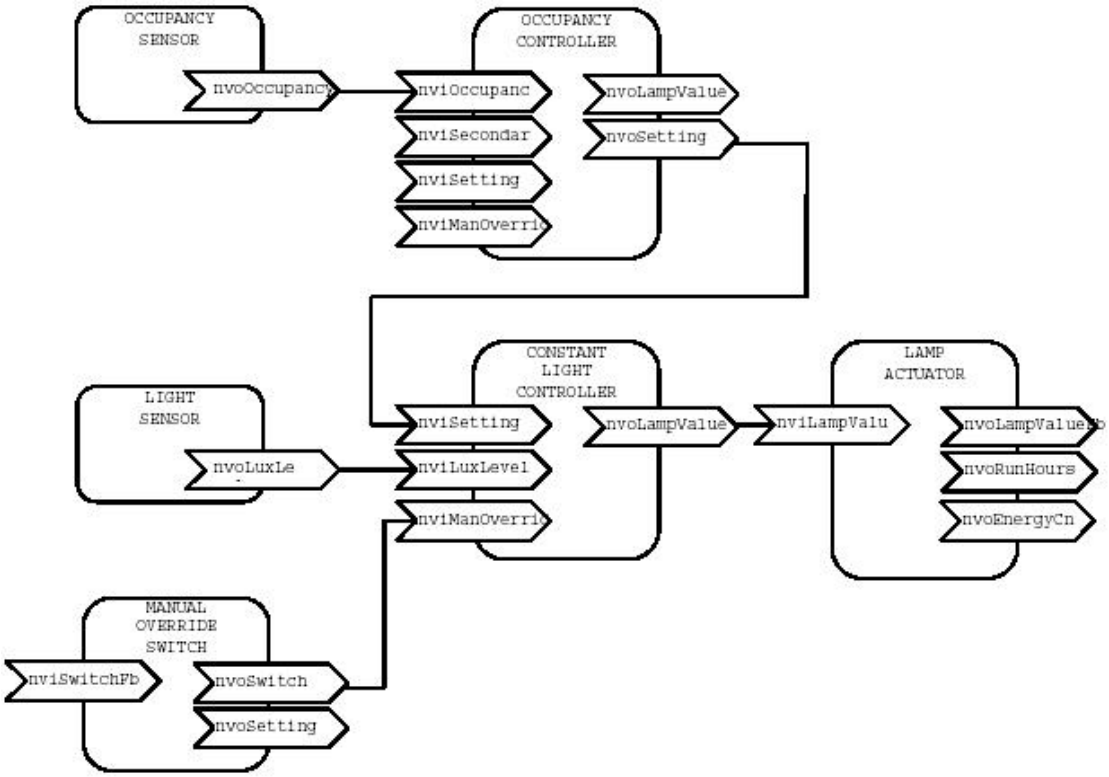


Fig. 5 Interaction between LONMARK Occupancy Controller Object, constant light controller, light sensor, lamp actuator and a manual override switch,

3. Management Level Integration

Above, we described an evolution of communications methods from proprietary, flat protocols to open, object-oriented information models. Simultaneous with this development, a parallel evolution has been taking place in the Information Technology realm – Web Services.

Web services are self-contained, modular applications that can be run over the Internet and can be integrated into other applications. Web services perform functions that can be anything from simple requests to complicated business processes. For

example, a weather bureau could offer a Web service that allows a building automation system to automatically retrieve temperature forecast data for use by various control algorithms. Similarly, the building automation system itself could offer a Web service that allows a tenant's accounting system to obtain up-to-the-minute figures on energy consumption. In the past, this type of data exchange would require a custom, "hard coded" data request to retrieve information that already existed in the host computer. A Web service, on the other hand, is a way to allow any authorized client to actually run an application on the host computer and generate data that didn't previously exist. In our accounting example, the tenant's computer would provide information on the inclusive dates and building areas, and the Web service host computer would calculate and return the energy consumption data.

These are very simple examples. In reality, Web services can be very complex, and a single application program may call upon multiple Web services. A Web service has to be able to combine content from many different sources. That may include furniture inventories, maintenance schedules and work orders, energy consumption and forecasts, as well as traditional building automation information. Web services have to serve all sorts of devices, platforms, and browser types, delivering content over a wide variety of connection types for a wide variety of purposes.

So, what is all about in Web services? Let’s first look at HTML (The Hypertext Markup Language). **HTML** format was designed for web pages to be read by humans. Like a universal word processor format, HTML combines text, pictures, and formatting information so a browser can display it on a screen. HTML is not adequate for information exchange between computers, however, because it provides no information about the data that may be contained on the screen and no way to search for specific pieces of data. For Web services to address all of these needs, two other, more flexible technologies are crucial: XML and SOAP.

XML (eXtensible Modeling Language) – XML is a technology for moving structured data across the web or a corporate network. Like the object oriented protocols described previously, XML documents include more than just raw data. An XML document includes a definition of the data structure, so the receiving computer knows what information is contained in which fields.

SOAP (Simple Object Access Protocol) – XML is basically a file format. SOAP is a way of using XML over a network. SOAP provides a computer application with a tool that can read the data definitions in an XML document and extract the required data. SOAP is to XML what HTTP is to HTML.

Let's look at a simple example using email service:

To: ssptang@vtc.edu.hk

From: tslam@vtc.edu.hk

CC:

Subject: Turn off the light

It's Friday. Don't forget to turn light off in B110.

When saved as HTML format:

```
<html>
    <title>Email</title>
    <body>
        <b>To:</b> ssptang@vtc.edu.hk<br>
        <b>From:</b>tslam@vtc.edu.hk<br>
        <b>CC:</b><br>
        <b>Subject:</b> Turn off the light<br>
        It's Friday. Don't forget to turn light off in B110.
        <br>
    </body>
</html>
```

When saved as XML format:

```
<email>
        <title>Email</title>
        <header>
                <to>ssptang@vtc.edu.hk</to>
                <from>tslam@vtc.edu.hk</from>
                <cc></cc>
                <subject>Turn off the light</subject>
        </header>
        <body>It's Friday. Don't forget to light off in
B110.</body>
</email>
```

In XML "to" is not displayable text, but a *tag* that identifies something; in this case a destination address. The data is: ssptang@vtc.edu.hk. The metadata (or data about the data) is what to do with ssptang@vtc.edu.hk. In this case use it as the destination of an email. E.g. the "to" field.

It is because of the <tags>, XML and HTML look similar, but they are used to accomplish very different things

- An HTML tag describes how text is to be displayed in a browser
- An XML tag describes the meaning of the text

A more complex file which taking energy data is shown below in XML:

```
<iLONDataLogger>
    <Log>
        <UCPTindex>0</UCPTindex>
        <UCPTfileName>/root/data/log0.dat</UCPTfileName>
        <UCPTstart>2002-10-30T01:15:00-08:00</UCPTstart>
        <UCPTstop>2002-11-10T17:45:00-08:00</UCPTstop>
        <UCPTlogLevel>36.0</UCPTlogLevel>

        <Element>
            <UCPTpointName>NVL_nvoPcValueDif_1</UCPTpointName>
            <UCPTlocation>pml1964</UCPTlocation>
            <UCPTlogSourceAddress>0.0</UCPTlogSourceAddress>
            <UCPTlogTime>2002-10-30T01:15:00-08:00</UCPTlogTime>
            <UCPTvalue>123.6</UCPTvalue>
            <UCPTunit>KW</UCPTunit>
            <UCPTpointStatus>AL_NO_CONDITION</UCPTpointStatus>
        </Element>
        <Element>
            <UCPTpointName>NVL_nvoPcValueDif_1</UCPTpointName>
            <UCPTlocation>pml1964</UCPTlocation>
            <UCPTlogSourceAddress>0.0</UCPTlogSourceAddress>
            <UCPTlogTime>2002-10-30T01:30:00-08:00</UCPTlogTime>
            <UCPTvalue>124.8</UCPTvalue>
            <UCPTunit>KW</UCPTunit>
            <UCPTpointStatus>AL_NO_CONDITION</UCPTpointStatus>
        </Element>
        <Element>
            <UCPTpointName>NVL_nvoPcValueDif_1</UCPTpointName>
            <UCPTlocation>pml1964</UCPTlocation>
            <UCPTlogSourceAddress>0.0</UCPTlogSourceAddress>
            <UCPTlogTime>2002-10-30T01:45:00-08:00</UCPTlogTime>
```

```
            <UCPTvalue>122.4</UCPTvalue>
            <UCPTunit>KW</UCPTunit>
            <UCPTpointStatus>AL_NO_CONDITION</UCPTpointStatus>
        </Element>
        <Element>
            <UCPTpointName>NVL_nvoPcValueDif_1</UCPTpointName>
            <UCPTlocation>pml1964</UCPTlocation>
            <UCPTlogSourceAddress>0.0</UCPTlogSourceAddress>
            <UCPTlogTime>2002-10-30T02:00:00-08:00</UCPTlogTime>
            <UCPTvalue>122.4</UCPTvalue>
            <UCPTunit>KW</UCPTunit>
            <UCPTpointStatus>AL_NO_CONDITION</UCPTpointStatus>
        </Element>
        <Element>
            <UCPTpointName>NVL_nvoPcValueDif_1</UCPTpointName>
            <UCPTlocation>pml1964</UCPTlocation>
            <UCPTlogSourceAddress>0.0</UCPTlogSourceAddress>
            <UCPTlogTime>2002-10-30T02:15:00-08:00</UCPTlogTime>
            <UCPTvalue>120</UCPTvalue>
            <UCPTunit>KW</UCPTunit>
            <UCPTpointStatus>AL_NO_CONDITION</UCPTpointStatus>
        </Element>
    </Log>
</iLONDataLogger>
```

All of the major Enterprise Software vendors are fielding products and platforms that support the Web services architecture using XML and SOAP, including Microsoft.NET, IBM WebSphere, Sun Microsystems SunOne, Echelon Panoramix, Hewlett Packard Web services platform, Oracle 9i, BEA Systems WebLogic, among others.

Since BACnet and EIB objects and LonMark functional profiles are information models, and XML is a modeling language, we could express these high level information models in XML and in so doing make them compatible with the emerging Web services architecture. Because of the flexibility of XML and the web services architecture, these high level models could be expanded to include other types of facility-related (but not necessarily building automation-related) information. If each building automation protocol developed its own XML model, however, we would have similar but incompatible system models. Today's problems of translating from

one protocol to another at the building controller level would become tomorrow's translation problems at the Web services level. What's needed is a unified system model, in XML, that can be used by any building automation protocol. One of the standardizing bodies is oBIX (www.orbix.org)

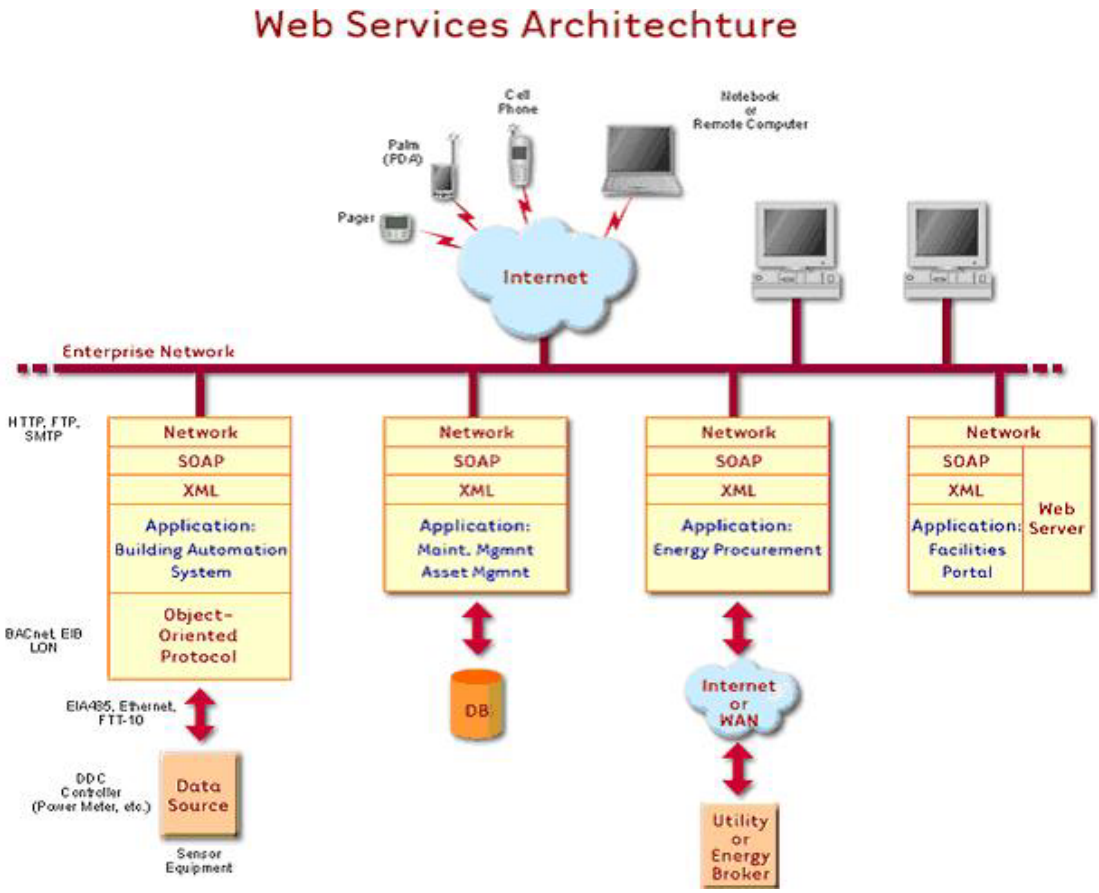


Fig. 6 Web services Architecture

It should also mention here that this push toward Web services architecture should not be interpreted as an end to standard protocols. Web Services are useful as a computer-to-computer or software-application-to-software-application interface, but they are "overkill" as a device-to-device interface. In order for interoperability to occur, vendors must not only agree on HOW they will communicate, but also on WHAT they will communicate. Because they include a high-level abstraction of what information is to be communicated, BACnet, EIB, and LonMark all provide the WHAT component of interoperability. By combining these information models with XML, and expanding the objective to include other non-HVAC related aspects of the facility, Web services can provide an information platform that is high-level, cross-platform, cross-discipline, and multi-vendor as shown in Figure 7.

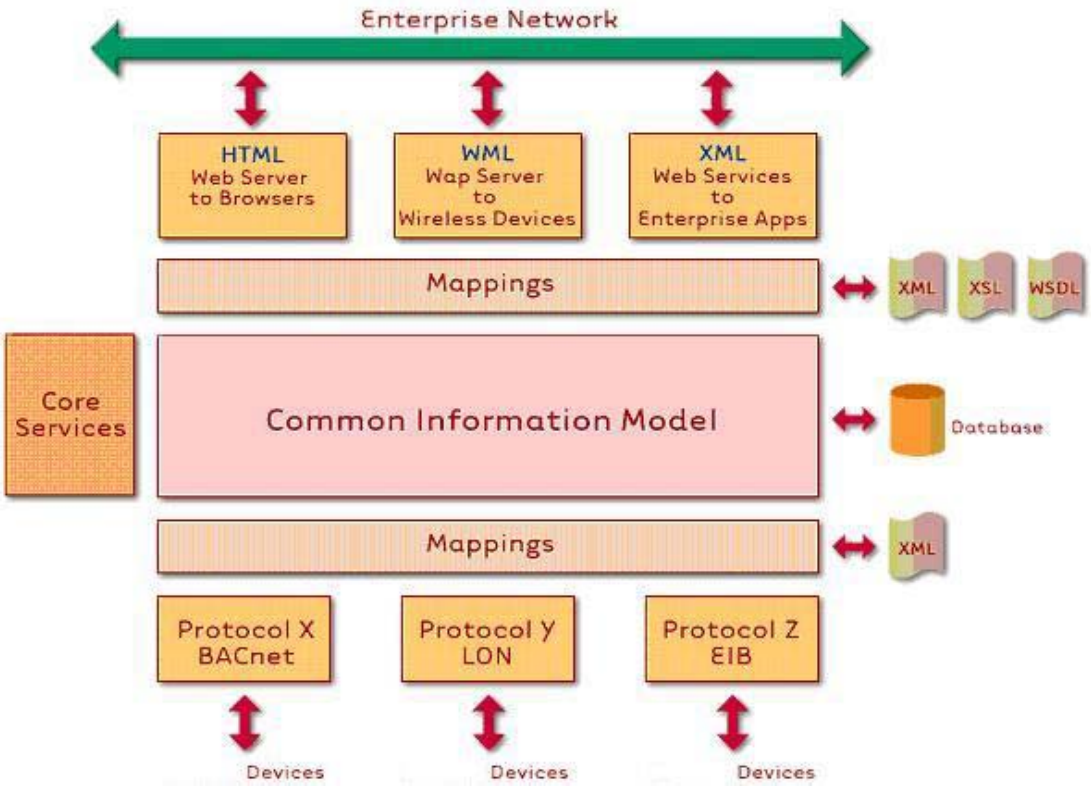


Fig. 7 Cross-platform, cross-discipline, and multi-vendor integrating platform

The model will include, but not be limited to, control and monitoring of HVAC, Fire Alarm, Security, Card Access, Asset Management, Energy, and other facility related systems and data sources. The document will be provided to standards organizations such as ISO, IAI, OASIS, ASHRAE and others for consideration as the basis for a formal international standard. The intent is to provide an interoperable standard which will enable systems to become interoperable. The following are the main sub-groups in this model:

1. The Data Modeling Group will need to determine how one model data that is to be exchanged between two devices or systems. Examples of attributes of the data model include: (a)Data type, (b) Units, (c) Name, (d) Accuracy, (e) Self documentation, etc.
2. The Services Group will need to create use cases and proposed services to deal with those functions that go beyond simple exchange of data. Examples of services include: (a) Notification of alarms, diagnostics and events, (b) Sharing of schedules and calendars, (c) Exchange of historical information such as trend logs.
3. To prevent user from intentionally or even inadvertently view or change critical facility information, the Security Group is to define the following: (a) Authentication. Verify that a user of data is allowed

access to perform the requested functions. Verify that the user is who they say that they are. (b) Encryption. Provide a method to prevent malicious users from viewing data as it moves across the network.

4. The Management Group needs to define how to configure, validate, and maintain a network of interconnected devices. Many of these functions are part of standard networking (for example DHCP, DNS, etc.). Other functions are more specific to facility management and real time control. Examples of areas to be defined by this group include: (a) Discovery of facility management devices, (b) Discovery of available data objects or variables, (c) Discovery of supported services, (d) Status of a device or sub-system (e.g. heartbeat), (e) Device / system backup and restore.

To this end, I should mention about OPC (OLE for process control), which was appeared on pg.3 in this chapter. The purpose of OPC is to provide a standard-based infrastructure for the exchange of process control data, no matter these data sources are coming from PLCs, DCSs, databases, RTUs and other devices. This data is available through different connections such as serial, Ethernet, or wireless. Different operating systems like Windows, UNIX,, etc. can work with OPC as OPC clients or OPC servers, if they use Microsoft's COM and DCOM technology to enable applications to exchange data on one or more computers using a client/server architecture as shown in Figure 8, with Figure 9 to illustrate drivers being need to be written for different processes.

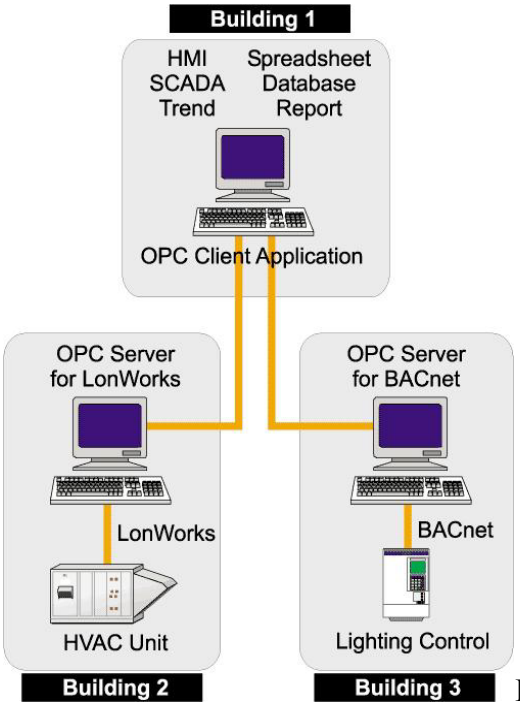


Fig. 8 OPC Architecture

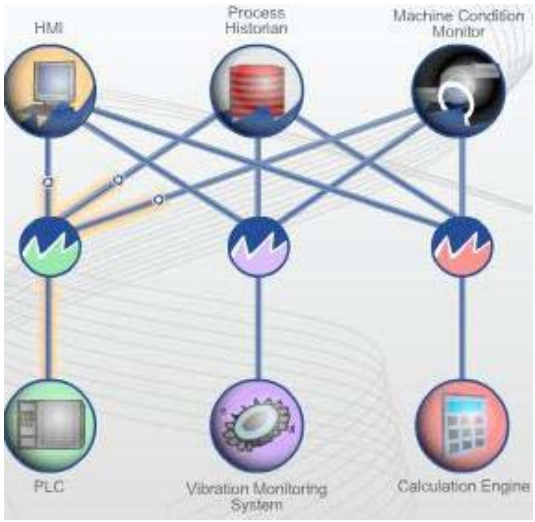


Fig.9 OPC Driver interfaces needed.

OPC defines a common set of interfaces, a simple example is shown in Figure 10. So applications retrieve data in exactly the same format regardless of whether the data source is a PLC, DCS, analyzer, software application or anything else. As a result, OPC is as an out-of-the-box, plug and play communication solution if OPC drivers for each manufacturers' devices are provided.

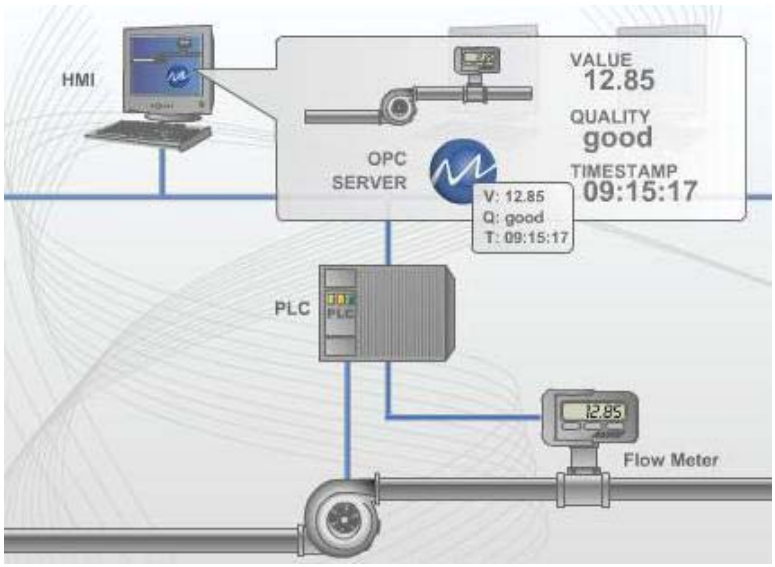


Fig. 10 OPC interface (*Value, Quality, Timestamp)*

OPC is not as flexible as XML that we have just mentioned, since at least manufacturers have to provide its own set of driver to the other manufacturers' devices, and also OPC is more for process control, and is not originally targeted for broader application such as enterprise network. Recently, OPC has come out with other specifications like OPC XML, OPC Security, OPC for Complex data, and OPC for ERP systems. Therefore, we may expect more functionality of OPC in the future.