The Hong Kong Polytechnic University

Department of Building Services Engineering

Development of An Intelligent Building Integration Platform Based On Web Services Middleware Technologies

Xu Zhengyuan

A thesis submitted in partial fulfillment of the requirements

for the Degree of Doctor of Philosophy

December 2008

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____ (Signed)

_____Xu Zhengyuan_____ (Name of student)

Department of Building Services Engineering

The Hong Kong Polytechnic University

Hong Kong, P. R. China

December 2008

ABSTRACT

Abstract of thesis entitled: Development of An Intelligent Building Integration Platform Based on Web Services Middleware Technologies

Submitted by: Xu Zhengyuan

For the degree of: Doctor of Philosophy

at The Hong Kong Polytechnic University in Dec., 2008.

The intelligent building integration platform presented in this thesis (named as IBmanager) employs standard communication protocol and distributed computing technologies, including object-oriented programming, data-subscription & event-driven technology, Web Services, XML driver technology, and value-added services plug-ins technology，to realize data and services integration and interoperation among distributed BASs on the Intranet/Internet.

The designed integration platform - IBmanager acts as an autonomous, self-contained unified integration unit (UIU). The IBmanager can work standalone as a full-function BMS, or just as a part of large-scale BMS applications. The distributed IBmanager installations can be deployed at chain-type architecture.

The IBmanager is designed consisting of a batch of function objects, allowing the systems to have good extensibility. The Data Point Objects are the core components of the overall platform, they drive the high-level functions running. Data-subscription & event-driven method is greatly used for the communication between function objects. Only the subscribed objects can trigger the events specified in the subscription process, and only the subscribers will handle the events they subscribe.

The bi-directional communication method is designed based on Web Service, this makes the transportation of COV (change of value) and A/E (alarm/event) messages based on Web Service feasible. The Web Service technology is used for the communication between the IBmanager and the driver, the communication between the IBmanager and its client, and access process to the remote database. Hence Web Service communication becomes the sole communication interface among IBmanager installations, remote database, drivers and their clients. The IBmanager can integrate other Web Services, meanwhile provides Web Services interfaces which can be invoked by other IBmanager installations, or other client applications.

Various sub-systems/devices can be integrated into the IBmanager with the designed XML Driver technology. Based on this technology, data from different protocols can be converted to common XML-encoded

messages. The sub-systems/devices can be anywhere on the Internet since the XML-encoded message can be transported by Internet standard protocol - HTTP. This expands the distributed driver communication beyond the LAN.

The remote heterogeneous databases can be integrated by Web Service. Every remote database is wrapped as a Web Services provider. Based on the design of Database Agent Object, a unified database access interface for the remote heterogeneous databases is presented.

With the help of the powerful communication capability of Web Services on heterogeneous platform and on the Internet, the middleware technology becomes a seamless integration platform on the Internet. With the support of the XML Driver and the Web Services interfaces presented, the middleware platform might be used to integrate other services, for example, weather report service. HVAC systems can optimize their control according to the data obtained from Web Services of the (government) Weather Bureau. Maintenance application can read specification data and manual of devices from manufacturers. Meanwhile the IBmanager middleware platform might be integrated by other applications on the Internet. For example, FDD (Fault Diagnosis & Detection) applications can read real time data and historical data from the IBmanager. By the same way, ERP (Enterprise Resource Planning)

application can obtain the energy consumption of the entire enterprise from the IBmanager. It ensures integration and interoperability among diverse facility systems/devices by connecting them to each other, to enterprise systems, and to the Internet in real time mode. This allows personnel using a standard web browser to measure, manage, and control a wide variety of energy, building, and security applications from anywhere in the world.

The system performance has been discussed and measured primarily. The communication latency resulted from drivers and client communication has been discussed, the roundtrip time of the IBmanager and the client has been measured.

As a common integration platform, users can develop their own applications based on the IBmanager. A practical HVAC optimization application is on-going on a big-scale project - the International Commercial Center (ICC) project. The IBmanager provides integration and value-added services platform support for this project.

ACKNOWLEDGEMENTS

I would like to express my sincerest appreciation to Prof. Shengwei Wang, my supervisor, Prof. Jiannong Cao and Dr. Mingli Chen, my co-supervisors for their invaluable guidance, patience and continuous support during the course of this research.

My special thanks go to Sun Hung Kai Properties Limited for their providing funding to support a practical project - International Commercial Center (ICC) to apply the research output. My special thanks also go to all other participants of the ICC project, from Takasago Thermal Engineering (H.K.) Co., Ltd, J. Roger Preston Limited, Johnson Controls Hong Kong Limited, for the benefits received from the exchanges and communications with them when applying the research output on the ICC site.

I would like to thank Mr. Siu Lo for his cooperation on supporting programming in the early stage of developing the integration platform. I also want to thank the colleagues in the department for their sincere care and help. The long road to accomplish this research is memorable because of their company.

Finally, I would like to express my deepest appreciation to my family

members: my father (although he left me two decades ago, his love and education has always been benefiting me in all my life), my mother, my honey daughter, especially my wife Yanping Huang for their unconditional trust and support in my life. This thesis is to them.

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION

1.1 Motivation

Nowadays there is a huge installation base of building automation system (BAS). These installations are compliant with various technologies, open or proprietary. They can be at the very beginning of their life cycles, the very end, or anywhere in between. Every installation represents a significant investment that the customer has already made. Unfortunately, some installations are getting obsolete as they have dedicated front-end interfaces and little integration to other systems. It is important to recognize that this investment must be protected, while at the same time allowing the customer to take advantage of today's open and standard technologies. In addition, there are many customers who have multiple systems, protocols and data format that they have to maintain. Although these multiple systems are declared open systems, they are different systems. In today's competitive world, this presents an exceedingly heavy burden for the owner, to train operating personnel and maintain their knowledge levels on multiple systems [3].

Ethernet/Internet connectivity, Web browser support, remote management services, connecting with back-end business applications – customers are demanding more advanced capabilities from today’s building intelligence. The ever-lasting developed IT industry is providing technology source for intelligent building (IB) industry. In particular, the broad acceptance and ever lowering cost of

Ethernet/Internet/XML (eXtensible Markup Language)/Web Service communications is finding its way into IB industry [1].

Operators/managers don't satisfy with the basic functions of building automation (BA) system any more, they continually present some new requirements for high-level value-added services. These value-added services include high-level energy-saving control strategy, FDD (Fault Diagnosis & Detection), decision analysis, data-mining and others. These services can make the facilities and buildings work with better performance and more automatic functions.

Nowadays, BAS is gradually recognized as a part of enterprise application. Enterprise systems are making a huge impact on corporations and other types of non-profit organizations. These enterprise systems include business financial systems, CRM (customer relationship management), human resources, and supply chain management. Some companies as well as consulting groups are providing integration services to make organizations very efficient. Buildings and facilities are now a significant area for organizations to include in such enterprise systems, corporations now appreciate that the effectiveness of their facilities can make a huge difference to their financial reports [2].

In conclusion, broader integration of BASs as Figure 1.1 is necessary to be developed. The solution should allow operators/managers to configure and manage equipments remotely and connect different systems and applications together via the Internet, thereby lowering support costs and acquiring broader information. It should integrate with enterprise applications and accommodate value-added services to

enable more complicated service offerings, such as predictive maintenance, performance analysis, and comprehensive reporting. For achieving this, there exist two challenges to be overcome. One is the incompatibilities and limited opportunities for the integration and interoperation of BASs from products of different vendors. What is needed is not only integrating the heterogeneous information together, but also making these diverse products communicate and interoperate together acting as a united system. The other is how to integrate BASs with enterprise applications and provide value-added services via the Internet. The integrated BAS will not only provide real-time information, but historical data for some complicated analysis and calculation of enterprise applications or value-added services. Meanwhile, the high-level applications (for example, a portal application) can supervise all the integrated BASs. These two challenges have frustrated real estate developers, building owners/operators, consultants and system integrators for many years.

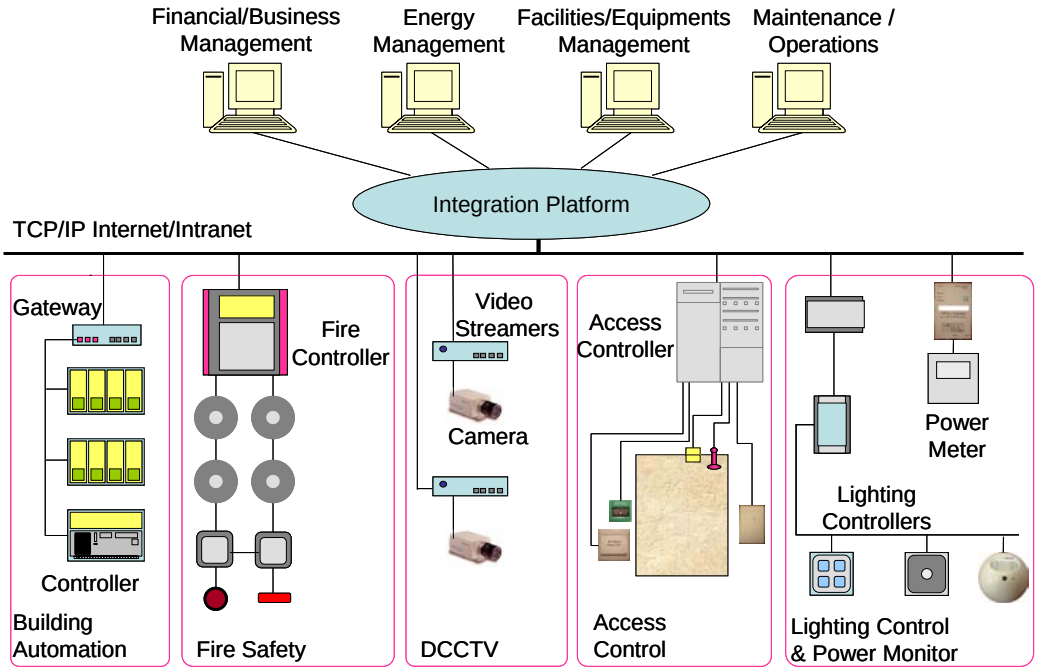


Figure 1.1 Concept of broader BAS integration

1.1.1 Need of Integration of Various Communication Protocols

In a typical BAS setup, different communication protocols are employed, even the products from the same syndicate. The products compliant with these different communication protocols cannot communicate together directly. *A popular way* to integrate the products compliant with various protocols is to employ hardware gateways [4]. The hardware gateway plays the role to convert a protocol to another protocol by mapping data points from one protocol to another protocol [5]. However, the development of the hardware gateway requires significant efforts and the developer needs to understand the technical details of the two protocols for conversion. Considering so many protocols existing currently (mainly proprietary protocols) and protocols-specific characteristic of gateway, the development of gateways is very costly. A lot of configuration efforts in the field are needed for the gateway to map the data points correctly when field engineers commission the system. This makes gateways expensive and complex to use [6]. The gateway also slows down the response due to the time required for the conversion. Furthermore, one can hardly program and configure a controller through a gateway [22]. *Another approach* (maybe the best approach) is to try to employ open and standard communication protocols to uniform the communication process from bottom layer to top layer. Several open solutions have become available in recent decades. One such solution is provided by BACnet: The Building Automation and Controls Network [7]. BACnet is a standard for data communication in building automation and controls systems that has been developed by the American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE). In 1995, BACnet was also adopted by ANSI (American National Standards Institute), and is now an American National Standard (ANSI/ASHRAE 135-1995) and ISO Standard (ISO 16484-5) [8]. Another

completely different solution is called LonWorks [9]. Various vendors have used LonWorks successfully in recent years to provide solutions for controls systems applications, in some cases involving multiple vendors [10]. In the recent years, great progress has been made on open standard communication protocols of BAS. BACnet has been developed as a full-fledged international standard, which can be used from bottom layer (field layer) to top layer (management layer) for intelligent building application. In the period of this study, ASHRAE has issued BACnet Web Services Interface Specification for Public Review [11]. The LonWorks camp has accommodated part of their products with Web Services interface as well [12].

However, proprietary protocols still dominate the current BAS market even today mainly due to business reasons according to the Frost and Sullivan Report A143-19, namely, "North American Building Automation Protocol Analysis" [13]. As a result, the endeavor to solve the integration and interoperation problem from bottom layer to top layer using a common protocol seems to have a long way to go. So the integration and interoperation of various communication protocols in the management layer is a beneficial and the last resort when the integration/interoperation on automation/field layer is not achieved. By this way, the information from various communication protocols is converted into a kind of common information model.

1.1.2 Need of Accommodating Value-added Services

Nowadays, operators/managers are gradually not satisfied with the basic building automation functions, they show great interest to present new ideas to make building “smarter”. They want that their own value-added services can be provided or developed based on the basic BAS. The services may be various, for example, the

acquiring of real-time weather broadcast information, the global pre-optimization of the total HVAC (Heating, Ventilation and Air Conditioning) system in a building, the control strategy of HVAC system. The latest services may be the data mining and decision-analysis [14], which can extract valuable data to provide decision support from a great volume of seemingly-useless data.

In order to fulfill these requirements, BAS must present an open and standard data exchange method for these value-added services. BAS acts as a data source and executor/distributor of output of the value-added services. This requires that they both must have a common data model to share data, and be designed to have good mechanism to cooperate together. Thus, when the optimization service reads and analyzes the data from BAS, then returns the optimization parameters to the BAS, thus BAS can make the facilities run according to these optimization parameters for better performance instantly.

1.1.3 Need of Integration with Enterprise Applications

During the past years, BASs have been increasingly seen as part of a much larger information system. Facility managers now routinely resort to specialized software for managing their tenant spaces, their assets, their equipment maintenance, and even for their energy procurement [17]. Facilities owners today are looking for a BAS as a data source to help them better run their business based on the infrastructure of their existing Intranets, Internet and the same standards as other IT (information technology) devices [1]. The requirements for "information integration" are now much broader

than those in the past. Broader “information and services” integration, more powerful functions integration among BASs, and even integration with other enterprise applications on the Internet become more and more important today [18]. Let’s take campus scheduling as a clear example. A system that can reserve a meeting room can automatically schedule the environmental, lighting, and security systems to adjust themselves on the basis of knowing when the room will be in use, and all this information is tied into the corporate scheduling, telephone conference, A/V resources and so on [2]. Other examples include room scheduling in hotels that would allow heating and cooling to be controlled based on whether a guest has checked in or checked out, controlling the HVAC and lighting in a college classroom based on the room's teaching schedule, and any other high-level business function that needs to interact with building systems [51].

An apparent trend in BAS industry is to enable the mechanical and electrical control systems in buildings to communicate with enterprise applications, and to provide a platform for developing new classes of applications that integrate control systems with other enterprise functions. Enterprise functions include processes such as Human Resources, Finance, Customer Relationship Management (CRM), and Manufacturing [19].

1.1.4 Necessity of Integration Platforms Development

However, current BMS is not good enough to meet the requirements mentioned above. So based on the necessity of integration with enterprise system/value-added

services and the situation of diverse communication protocols in the market, developing a middleware platform to achieve integration among BASs on higher level, namely, management layer is very useful. This middleware platform will focus on integration/interoperation on management layer and data-exchange with value-added services/enterprise applications. It will not substitute or influence directly the manufacturer-specific products, protocols, and configuration tools. That means, on the automation layer and field device layer, the control networks can keep untouched, running as the original architecture and with the original protocols, the controllers can be configured by the original configuration tools from the manufacturers.

The fast-developed standard IT technologies can provide ideas and support for the building automation industry greatly. In today's IP centric world, one can expect real-time access to information in order to make informed, intelligent and strategic business decisions. Based on the integration platform to be designed with the latest IT technologies, one should expect the same from the building automation system.

This integration platform will accommodate the heterogeneous sub-systems compliant with diverse communication protocols, provide a unified data model and interface for integration/interoperation applications and value-added services. By this platform, the sub-systems/field devices are not only integrated to provide real-time and historical data information to people and the enterprise, but interoperate each other. With the standard open technologies, this platform will communicate with enterprise system and be thought as part of the enterprise system. Figure 1.2 shows the overall architecture of Intelligent Building platform to be developed.

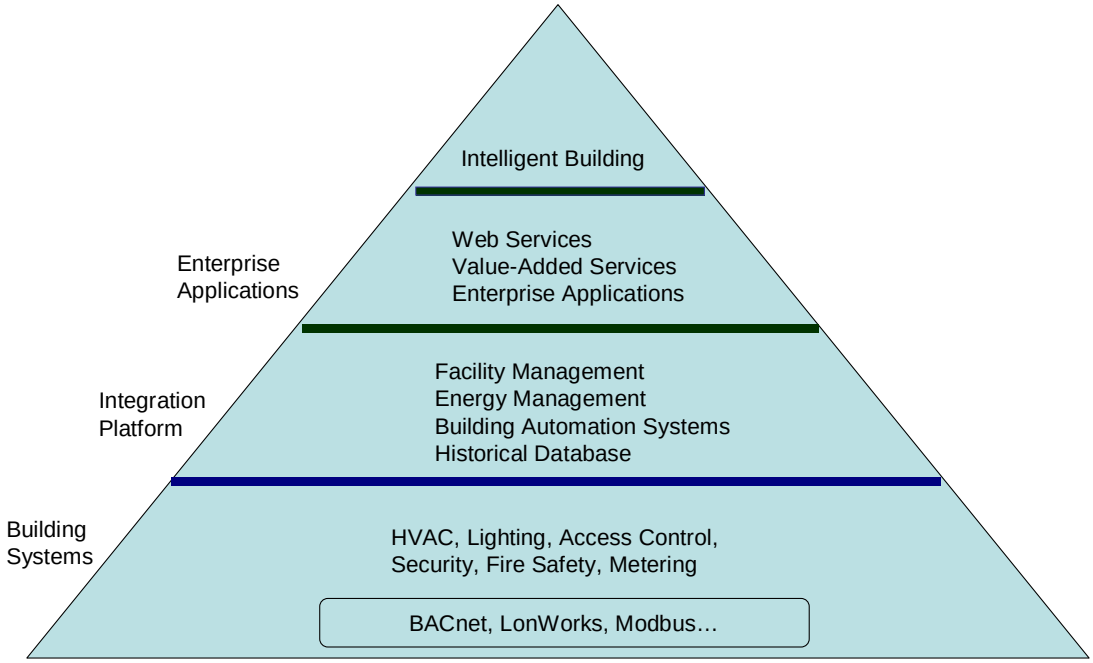


Figure 1.2 Overall architecture of Intelligent Building platform to be developed

1.2 Literature Review

In an intelligent building, there install a great deal of devices. These devices need to communicate with each other as well as with data acquisition and processing systems. The data acquisition and processing systems will automate the communication and interpretation of the mass of data for various value-added services. It is known as M2M (Machine-to-Machine) – connecting machines to people, each other and the enterprise. The focused issues include how these machines communicate, how they are managed, and most importantly, how the world (humans, businesses, and society) interacts with them in an open manner almost anywhere in the world [21].

In some modern complex with multiple sub-systems from various manufacturers installed, these sub-systems are supervised via their own BAS software usually. These

BAS software from different manufacturers may be based on various hardware platforms and OSs, providing different communication interfaces. The integration software will communicate with the various BAS software or field devices directly. In both cases, the integration software must make various BAS software/devices communicate effectively to realize interoperation/integration among various sub-systems and enterprise applications. These will relate to the conversion of disparate communication protocols and data format. The evolution of communication protocols and data manipulation technologies will be reviewed later.

The definition of system integration has been in the continual flux from the scope and depth. The architecture of integration system has been always in evolution as well. The architecture of IT industry has greatly evolved in the past decades. The IB industry can learn a lot from IT industry. The distributed architecture makes the different parts/components of the integrated systems (including devices, driver, server, and client) to work together flexibly. The middleware technologies are evolved to facilitate the development of distributed computing programming. Several different kinds of middleware technologies has been developed and popularly used in IT industry. So the system integration, its architecture and middleware technologies used will be reviewed as well.

1.2.1 Definition and Classification of System Integration

Systems integration is the process of connecting building systems together to provide a common user interface and also to achieve functionality between systems. Examples of systems integration are connecting all of the HVAC equipment together into a cohesive system or connecting together card access, lighting control and HVAC

so that when an employee swipes their card after hours the lights and HVAC come on. The connected card access, lighting control and HVAC may be compliant with different communication protocols from different providers. [24]

i). Primary System Integration

According to McGowan J., system integration is defined in very distinct ways based upon whether the focus is on control or interface communication. The basic definition for system integration with focus on control has traditionally been the process of achieving control interaction between the sequences for fire, security and DDC for HVAC [25]. This is a primary integration way by wiring the I/O (input/output) to the same control system and making the logic sequences among the I/O points. This method has phased out gradually today.

ii). Interoperability System Integration

In the integration case with focus on interface communication, communication must be addressed to accomplish some form of data interchange between existing or "legacy" systems. This requires "drivers" or specialized software packages that translate between languages to allow communication between legacy and new open systems from one point of interface. This may be the most common type of system integration occurring today, and can also come to interoperability, which allows for seamless communication, or interface interoperability, between systems [25]. In this stage buildings are always considered to be operated in a largely standalone manner.

iii). Enterprise Application Integration

As building connectivity technology matures and increasingly adopts IT-based solutions, the building is fast becoming another component of today's enterprise systems. This provides corporations with an advanced level of facility control as a result of new, important information previously unavailable without assistance of the IT network [26].

The interaction applications of buildings and the enterprise include energy management, business process interaction, efficiency improvement, as well as core functions of facilities, all in order for facilities to house the corporation's business activities. For example, an enterprise integration may look at all of the energy use from a group of buildings in several cities, read utility rates in real time, and then place all of the buildings into a particular energy saving mode based on the rate. A second example would be bringing critical building parameters up to the enterprise then using that information to make maintenance decisions. What we observed is that the integration of building systems is one of the last portions of business to be connected. Other critical business functions such as human resources, finance, sales and marketing, and manufacturing have been connected to enterprise applications as a matter of course [27].

1.2.2 Evolution of IB Communication Protocols

Beginning in the 1980's, more and more manufacturers of HVAC-related equipment began incorporating microprocessor-based controllers in their products at the factory. In the beginning, these controllers were designed to be stand-alone. With

the increase in popularity of networked building automation and energy management systems, communications ports were added to these stand-alone controllers and various communications protocols or languages evolved. The earliest of these protocols were often proprietary to the equipment manufacturer. Later, several of the most common protocols became de facto standards, for example Modicon's MODBUS [96]. Other companies recognized the need for a common protocol as a business opportunity and developed their own protocols together with supporting products to supply to the industry (Echelon's LonWorks, Bosch's CAN [97], etc.). Eventually, several industry standards bodies formed committees to define protocols that would be available to be deployed without licenses or royalties (ASHRAE-BACnet [8], OPC Foundation-OPC [99], Profibus International-Profibus [98], etc.) [17].

Basically, the protocols can be classified into two categories: protocols with flat data structures and protocols with object-oriented data structures as Figure 1.3 [17]. Initially these standard protocols utilized flat data structures of independent and possibly unrelated values (MODBUS registers, LonTalk Network Variables, OPC Items, etc.). In a flat data structure, each piece of information stands alone. For example, 76.6 might represent a temperature, but the units and name of that data sample would each be stored separately in the program. As object-oriented programming paradigms gained widespread acceptance as an alternative to flat data structures, a more object-oriented approach toward field data was desired. In an object-oriented data structure, 76.6 would be packaged with the units (℃) and the

name (Zone Temperature). Having these related pieces of information stored together as an object assists in data interpretation. In addition, multiple objects can then be grouped together into another object in a hierarchical fashion. For example, the Zone Temperature and Zone Setpoint objects might be grouped into a Zone Control object. This movement toward object-oriented programming gave birth to the current generation of object-oriented protocols - BACnet, LonMark Functional Profiles [100], and European Installation Bus Object Interface Specification (EIB - ObIS), among others [17].

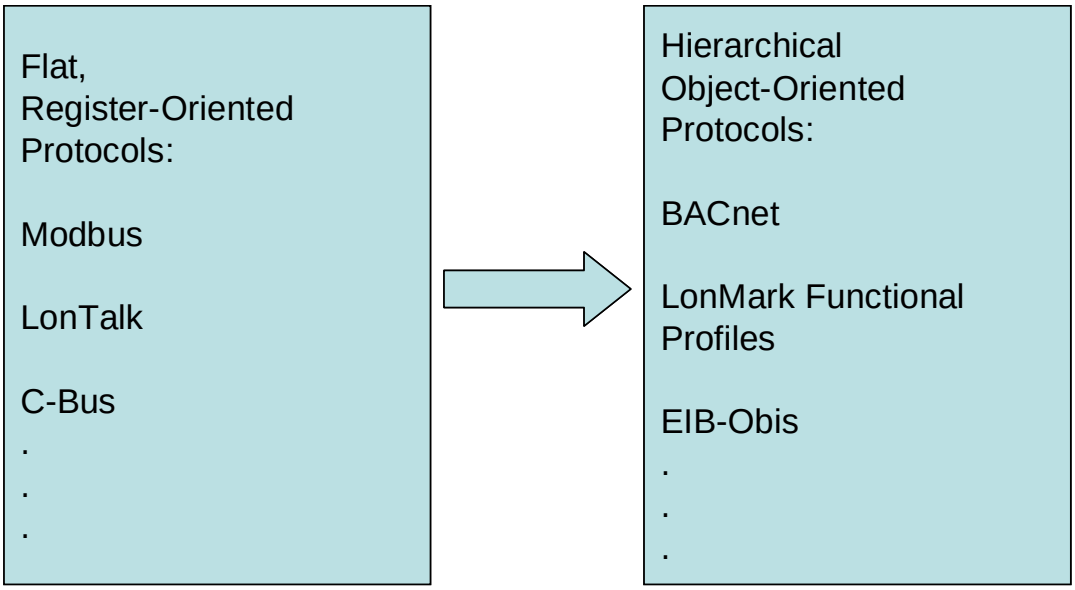


Figure 1.3 BAS protocol evolution

In the 1980's, there wasn't a protocol that could meet the majority of a Building Automation system's needs. As it goes into the 21st century, the problem is not one of "not enough" but one of "too much." 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 bridging more than two can become unwieldy [17].

In order to be compatible to the various standard protocols, the integration platform must be made to communicate various products from different vendors or compliant with various protocols. Various protocols should be connected and their different data formats are interpreted to common data format together.

1.2.3 Evolution of IT and Its Influence on IB

Above, evolution of communications methods from proprietary, flat protocols to open, object-oriented information models is described. Simultaneous with this development, a parallel evolution has been taking place in the Information Technology (IT) realm. The 1970's were the years of the mainframe/dumb terminal architecture. The 1980's saw the birth of the PC. The 1990's brought networked PC's with client-server architecture. In the late 90's and early 00's, (due largely to the explosive popularity of the Internet) we're seeing a return to the 70's style approach relabeled the "thin-client architecture." Dumb terminals have been replaced with web browsers. Mainframe computers have been replaced with Web server/Web Services farms as Figure 1.4 [17].

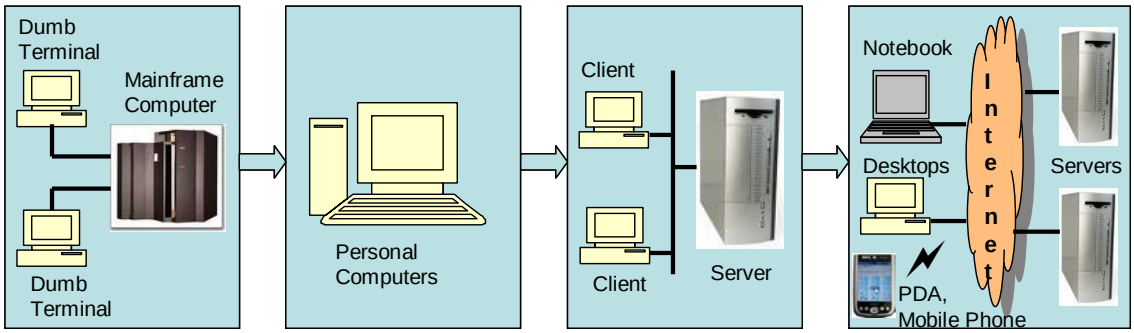


Figure 1.4 Evolution of Information Technology

In their white paper “Web Services - The Web's next revolution”, IBM points out several trends that are becoming apparent [17, 92]:

- Content is becoming dynamic – There are a lot of 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. Today, building integration software may be required to provide up-to-the-minute content, incorporating news headlines, weather forecasts, and current chilled water supply temperatures on the same page.
- Bandwidth is getting cheaper - Some analysts predict that in a few years, there will be enough bandwidth for a full-motion video channel that records the life of every person on earth. Whether that happens or not, bandwidth is growing exponentially cheaper year after year.
- Storage is getting cheaper - The capacities of hard drives, DVDs, CD-ROMs, and removable storage media are far greater than they were a few years ago.

This trend is likely to continue.

- Enterprise computing is becoming more important - The need to integrate information from our familiar desktop PCs with mobile phones, pagers, and palmtop computers on the low-end and with mini and mainframe-based corporate information systems on the high-end is increasing.

The evolution of IT has been influencing the development of intelligent building industry greatly. The latest IT technologies are getting more applications in the IB industry sooner than before. As a result, the IB industry has been going to IP technologies and Web applications as well.

1.2.4 Traditional Middleware Technologies and Its Applications in IB Integration

(i) Middleware Concept

Middleware is a class of software technologies designed to help manage the complexity and heterogeneity inherent in distributed systems. It is defined as a layer of software above the operating system but below the application program that provides a common programming abstraction across a distributed system, as shown in Figure 1.5 [29]. In doing so, it provides a higher-level building block for programmers than Application Programming Interfaces (APIs) such as sockets that are provided by the operating system. This significantly reduces the burden on application programmers by relieving them of this kind of tedious and error-prone programming. Middleware is sometimes informally called “plumbing” because it connects parts of a distributed application with data pipes and then passes data between them [29].

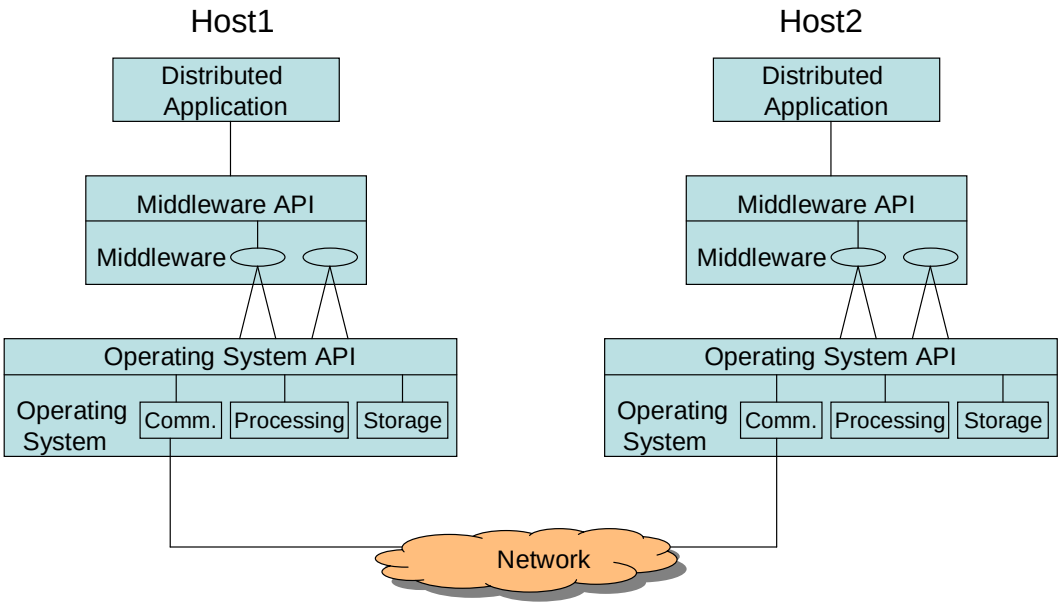


Figure 1.5 Middleware layer in context

(ii) Traditional Middleware Technologies

The ability to construct applications using objects from different vendors, running on different machines, and on different operating systems, it is not an easy task. The need for interaction between the software objects led to the specification of middleware models. The most widely-publicized middleware initiatives, including COM/DCOM (Component Object Model/Distributed Component Object Model) [30], CORBA (Common Object Request Broker Architecture) [31], and JAVA/RMI (remote Method Invocation) [32] have been developed to facilitate the communication among distributed applications.

CORBA

CORBA is short for Common Object Request Broker Architecture, an architecture that enables pieces of programs, called objects, to communicate with one another regardless of what programming language they were written in or what

operating system they're running on. CORBA was developed by an industry consortium known as the Object Management Group (OMG).

CORBA relies on a protocol called the Internet Inter-ORB Protocol (IIOP) for remoting objects. Everything in the CORBA architecture depends on an Object Request Broker (ORB). The ORB acts as a central Object Bus over which each CORBA object interacts transparently with other CORBA objects located either locally or remotely. Each CORBA server object has an interface and exposes a set of methods. To request a service, a CORBA client acquires an object reference to a CORBA server object. The client can now make method calls on the object reference as if the CORBA server object resided in the client's address space. The ORB is responsible for finding a CORBA object's implementation, preparing it to receive requests, communicate requests to it and carry the reply back to the client. A CORBA object interacts with the ORB either through the ORB interface or through an Object Adapter - either a Basic Object Adapter (BOA) or a Portable Object Adapter (POA). Since CORBA is just a specification, it can be used on diverse operating system platforms from mainframes to UNIX boxes to Windows machines to handheld devices as long as there is an ORB implementation for that platform. Major ORB vendors like Inprise have CORBA ORB implementations through their VisiBroker product for Windows, UNIX and mainframe platforms and Iona through their Orbix product [33].

DCOM

COM is a technology that Microsoft developed to replace OLE (Object Linking and Embedding) and DDE (Dynamic Data Exchange). This framework was defined by the combination of COM and OLE Controls (OCX). Distributed Component

Object Model (DCOM) emerged to address COM's shortcomings in supporting remote components. DCOM is an extension to COM that allows networked interaction between two programs even if they are written in different programming languages [34]. DCOM which is often called "COM on the wire", supports remoting objects by running on a protocol called the Object remote Procedure Call (ORPC). This ORPC layer is built on top of DCE's RPC and interacts with COM's run-time services. A DCOM server is a body of code that is capable of serving up objects of a particular type at runtime. Each DCOM server object can support multiple interfaces each representing a different behavior of the object. A DCOM client calls into the exposed methods of a DCOM server by acquiring a pointer to one of the server object's interfaces. The Session Object then starts calling the server object's exposed methods through the acquired interface pointer as if the server object resided in the client's address space. As specified by COM, a server object's memory layout conforms to the C++ vtable layout. Since the COM specification is at the binary level it allows DCOM server components to be written in diverse programming languages like C++, Java, Object Pascal (Delphi), Visual Basic and even COBOL. As long as a platform supports COM services, DCOM can be used on that platform. DCOM is now heavily used on the Windows platform. Companies like Software AG provide COM service implementations through their EntireX product for UNIX, Linux and mainframe platforms; Digital for the Open VMS platform and Microsoft for Windows and Solaris platforms [33].

Java/RMI

Java Remote Method Invocation (Java/RMI) enables the programmer to create distributed Java technology-based to Java technology-based applications, in which the

methods of remote Java objects can be invoked from other Java virtual machines, possibly on different hosts. Java/RMI relies on a protocol called the Java remote Method Protocol (JRMP). Java relies heavily on Java Object Serialization, which allows objects to be marshaled (or transmitted) as a stream. Since Java Object Serialization is specific to Java, both the Java/RMI server object and the Session Object have to be written in Java. Each Java/RMI Server object defines an interface which can be used to access the server object outside of the current Java Virtual Machine (JVM) and on another machine's JVM. The interface exposes a set of methods which are indicative of the services offered by the server object. For a client to locate a server object for the first time, RMI depends on a naming mechanism called an RMIRegistry that runs on the Server machine and holds information about available Server Objects. A Java/RMI client acquires an object reference to a Java/RMI server object by doing a lookup for a Server Object reference and invokes methods on the Server Object as if the Java/RMI server object resided in the client's address space. Java/RMI server objects are named using URLs and for a client to acquire a server object reference, it should specify the URL of the server object as you would with the URL to a HTML page. Since Java/RMI relies on Java, it can be used on diverse operating system platforms from mainframes to UNIX boxes to Windows machines to handheld devices as long as there is a Java Virtual Machine (JVM) implementation for that platform. In addition to Javasoft and Microsoft, a lot of other companies have announced Java Virtual Machine ports [33].

(iii) Weakness of Traditional Middleware Technologies

Weakness of DCOM and CORBA/IIOP:

DCOM and CORBA are both reasonable protocols for server-to-server

communications. However, both DCOM and IIOP have severe weaknesses for client-to-server communications, especially when the client machines are scattered across the Internet [35].

DCOM and CORBA/IIOP both rely on single-vendor solutions to use the protocol to maximum advantage. Though both protocols have been implemented on a variety of platforms and products, the reality is that a given deployment needs to use a single-vendor's implementation. In the case of DCOM, this means every machine runs Windows. (Although DCOM has been ported to other platforms, it has only achieved broad reach on Windows.) In the case of CORBA, this means that every machine runs the same ORB product. Yes, it is possible to get two CORBA products to call one another using IIOP. However, many of the higher-level services (such as security and transactions) are not generally interoperable at this time. Additionally, any vendor-specific optimizations for same-machine communications are very unlikely to work unless all applications are built against the same ORB product.

DCOM and CORBA/IIOP both rely on a closely administered environment. The odds of two random computers being able to successfully make DCOM or IIOP calls out of the box are fairly low. This is especially true when security is involved. While it is possible to write a shrink-wrap application that can use DCOM or IIOP successfully, doing so requires much more attention to detail than the typical sockets-based application. This is especially applicable to the unglamorous but necessary task of configuration/installation management.

DCOM and CORBA/IIOP both rely on fairly high-tech runtime environments. While in-process COM is simple, building the COM/DCOM remoting plumbing is definitely not an easy project. IIOP is a simpler protocol to implement than DCOM,

but both protocols have their fair share of complex rules dealing with data alignment, type information, and bit twiddling. This makes it difficult for the average programmer to simply make a CORBA or DCOM call without the benefit of an ORB product or OLE32.DLL.

Perhaps the most damning limitation of DCOM and CORBA/IIOP is their inability to work in Internet scenarios. In the case of DCOM, it is a great headache to configure and pass the domain-based authentication with your servers. Worse, if a firewall or proxy server separates the client and server machines, the likelihood of either IIOP or DCOM packets getting through is extremely low due to the HTTP bias of most Internet connectivity technology. While vendors like Microsoft, Iona, and Visigenic have all built tunneling technology, these products tend to be very sensitive to configuration mistakes and are not interoperable [36].

Although the disadvantages discussed above, DCOM or IIOP are still suitable to work within a server farm. All of the host machines in the server farm are under a common administrative domain, which makes consistent configuration quite likely. The relatively small number of machines also helps to keep the costs of using commercial ORB products under control, as a smaller number of ORB licenses are needed. Finally, it is likely that all of the host machines in a server farm will have direct IP connectivity, removing the firewall-related problems of DCOM and IIOP [37].

Weakness of RMI:

RMI was developed as a simple distributed-objects programming model for Java. Simplicity has its advantages, e.g. better performance, but it is also RMI's primary

weakness. JAVA/RMI has the disadvantages as below:

- Synchronous communication: calling process blocks until there is a response
- Tightly coupled: client must find recipients, and know method arguments
- No persistence
- A lot of connections can be difficult to scale [38]
- Java/RMI only supports Java
- Proprietary protocol by single vendor
- Requires RMI-lookup
- Requires non-standard port [39]

(iv) Applications of Traditional Middleware Technologies in IB Industry

Lu N. et al. [40], Wang Y. et al. [42], Zheng F. et al. [23] and Guo H. et al. [41] have made the primary design about CORBA used on IB system. Some may have been used in practical projects. However, no broader applications can be found. JAVA/RMI can be seen on some mobile IB applications and agent applications [32] and hasn't been used broadly in the IB integration because Java/RMI can only be used with the Java programming language.

Probably due to the broad use in the automation industries and the popularity of Windows platform, some manufacturers have applied DCOM-based OPC (OLE for Process Control) technology on BAS integration [43]. It seems OPC has broader use than the other middleware technologies mentioned above. In the beginning of this research, OPC is adopted as main technology for integration platform. However this choice was changed after practical trials. In this thesis, the OPC technology will be discussed later as part of this research as well.

1.2.5 Web Services and Its Applications on IB Integration

(i) Web Services

A new promising technology is Web Services. XML and Web Services technology is put forward by World Wide Web Consortium (W3C) organization [44]. It will help greatly integration and interoperation among applications over the Internet. SOAP is communication protocol for accessing a Web Service. As an emerging distributed middleware technology that uses a lightweight and simple XML-based protocol, SOAP allows applications to exchange structured and typed information across the Web. It is designed to support automated Web Services based on a shared, decentralized, and open web infrastructure. SOAP applications can be written in a wide range of programming languages (such as Java, C++, C, Perl, and C#), used in combination with a variety of Internet protocols and formats (such as HTTP, SMTP, and MIME), and can support many types of applications ranging from messaging systems to RPC (remote procedure call) [45].

Rather than competing with existing middleware technologies, Web Services is evolving into a role of integrating middleware. Unlike its RPC and EAI (Enterprise Application Integration) technology predecessors, Web Services technologies and standards enjoy unprecedented industry backing. All the major players and most of the minor ones fully support and endorse the standardization of SOAP and WSDL (Web Services Description Language). So far, the industry remains unfragmented with respect to these standardization efforts, something that never happened for Sun RPC, DCE, COM/DCOM, CORBA, or J2EE. Web Service comes to being able to integrate different types of middleware, including CORBA, J2EE, and Microsoft .NET [45]. It is even called “Middleware for Middleware”. The comparison of different middleware

technologies is conducted in Table-1.

Table-1 Comparison of different middleware technologies [46]

	Platform availability	Applicable to	Mechanism	Implementations
COM /DCOM	Originally PC platforms, but becoming available on other platforms	"PC-centric" distributed systems architecture	APIs to proprietary system	one
CORBA	Platform-independent and interoperability among platforms	General distributed system architecture	Specification of distributed object technology	many
Java/RMI	Where Java virtual machine (VM) executes	General distributed system architecture and Web-based Intranets	Implementation of distributed object technology	various
Web Services	Platform-independent and interoperability among platforms	General distributed system architecture and Web-based Intranets	XML and SOAP (Simple Object Access Protocol)	many

While traditional middleware platforms provide great implementation vehicles for services, none of them is a clear winner. The strengths of the Web Services as an information integrator and distributor, namely simplicity of access and ubiquity, are important in resolving the fragmented middleware world where interoperability is hard to come by. The Web Services complements these platforms by providing a uniform and widely accessible interface and access glue over services that are more efficiently implemented in a traditional middleware platform. Viewed from an n-tier application architecture perspective, the Web Service is a veneer for programmatic access to a service which is then implemented by other kinds of middleware. Access consists of service-agnostic request handling (a listener) and a facade that exposes the operations supported by the business logic. The logic itself is implemented by a traditional middleware platform as Figure 1.6 [47].

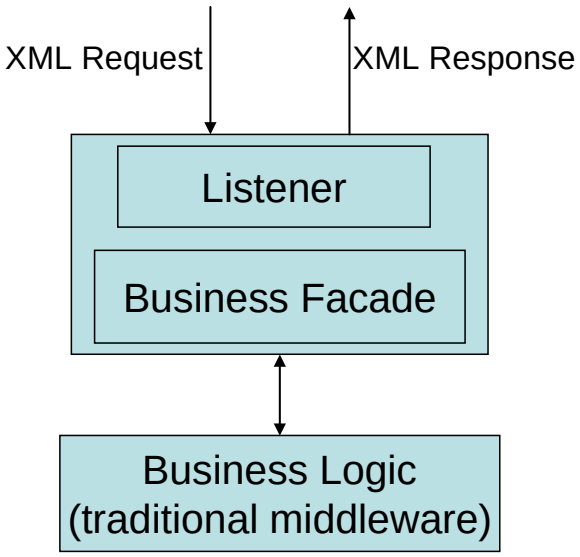


Figure 1.6 Generic Web Service architecture

(ii) Researches/Applications of Web Services on IB

As a new breed of Web application, Web Services is 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 an 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 [17].

Donlon M. claimed “Finally, using TCP/IP connections, protocols like XML will dominate the future of interoperability among embedded devices - even in building automation.” [48]. Craton D. and Robin E. appealed to construct information models based on Web Services [17]. They also claimed that “it might be possible to expose information as XML at a building-controller level, it would not be practical to do so at a zone or unitary-controller level.” At the duration of this research on-going, there are some communities engaged in the research of Web Services application in IB industry.

(i) oBIX

oBIX (Open Building Information eXchange) is an industry-wide initiative to define XML- and Web Services-based protocol to enable communications between building mechanical and electrical systems and enterprise applications. oBIX will instrument the control systems for the enterprise. oBIX was originally established as a working group within CABA (Continental Automated Buildings Association) [49], then the CABA Board of Directors announced that it has initiated a process to transfer governance of oBIX to a technical committee (TC) at the Organization for the Advancement of Structured Information Standards (OASIS) in April 28, 2004 [50]. OASIS is a global, non-profit consortium that focuses on the development and adoption of e-business standards [2]. This defined protocol will enable facilities and their operations to be managed as full participants in knowledge-based businesses.

The oBIX specification will utilize Web Services for exchange of information with the mechanical and electrical systems in commercial buildings. The oBIX TC proposes to develop a publicly available Web Services interface specification that can be used to obtain data in a simple and secure manner from HVAC, access control, utilities, and other building automation systems, and to provide data exchange between facility systems and enterprise applications. In addition, the TC will develop implementation guidelines, as needed, to facilitate the development of products that use the Web Service interface [66].

From the “Committee Specification 01 - oBIX 1.0” which is issued on 5 December 2006, we can find it specifies an object model and XML format used for machine-to-machine (M2M) communication [67]. However, its focus is only on the definition of standard interface, not the implementation of the integration platform.

(2) BACnet/Web Services

The BACnet/XML Working Group also is working on BACnet/Web Services, using XML to represent BACnet messages so that Web Services gateways can provide building data to enterprise systems across the Internet or an Intranet [28]. At ASHRAE's 2004 Annual Meeting, the Standard 135 committee proposed addendum 135-2004c for public review. This addendum defines a standard means of using Web Services to integrate facility data from disparate data sources, including BACnet (A Data Communication Protocol for Building Automation and Control Networks) networks, with a variety of business enterprise applications [8]. September 2006 this addendum is approved as part of formal standard [20].

The addendum provides ways to standardize and use XML in representing certain data types that are relevant to BACnet and defines the services as below:

getValue Service

getValues Service

getRelativeValues Service

getArray Service

getArrayRange Service

getArraySize Service

setValue Service

setValues Service

getHistoryPeriodic

getDefaultLocale

getSupportedLocales

Naturally the addendum is compatible with the BACnet protocol, but it is not limited to BACnet. Indeed, one of its most useful applications may be to serve as a standard for exchanging data between building automation systems using different protocols. Web Services could be an ideal way to make a “top end” connection between systems running BACnet, LonWorks, MODBUS, or any proprietary protocol. Engineers would not have to learn the details of each individual protocol to program the connections, they would only have to understand the Web Services standard. A Web Services connection would also avoid the problems with incompatible baud rates, wire types, proprietary communication chips, and all the other issues that can come into play when a gateway is used to connect dissimilar protocols [68].

1.2.6 Offline and Online Value-added Service

In the current real application project, the Value-added Service (for example, HVAC optimization application) is mainly restricted in off-line application. The current normal optimization software reads or imports data from the historical database of some commercial BAS software platforms. They calculate and make some suggestions for facility managers based on the historical data. However, the environment (for example, weather) may be changed fast, the offline optimization software cannot suggest and adjust the parameters timely. Thus, the parameters suggested by offline method cannot command the BAS directly for lack of in-time and flexible communication channel. The optimization is processed offline and needs the intervention of operator.

So online method is good method to be used to keep real-time data-exchange between Value-added Service and BAS. In Aug. 2008, Lawrence Berkeley National Laboratory (LBNL) was published to be developing an online system used to connect building simulation and BAS [101]. The published paper “describes the Building Controls Virtual Test Bed (BCVTB) that is currently under development at Lawrence Berkeley National Laboratory. An earlier prototype linked EnergyPlus with controls hardware through embedded SPARK models and demonstrated its value in more cost-effective envelope design and improved controls sequences for the San Francisco Federal Building”. “The BCVTB is a more modular design based on a middleware that we built using Ptolemy II”, which is Java based developed by the University of

California at Berkeley. “In future work we will also implement a BACnet interface that allows coupling BACnet compliant building automation systems to Ptolemy II”. “The BCVTB is a modular, extensible, open-source software platform that allows designers, engineers and researchers of building energy and control systems to interface different simulation programs with each other and, in the future, with Building Automation Systems (BAS)”. From the text above, we can see that the connection with BAS is still on development on the publish time.

Optimization calculation process needs integration of great volume of data and powerful CPU calculation capability. These will lead to remarkable calculation delay. So there are some issues to be solved in online method.

1.2.7 Conclusive Remarks

From the literature reviewed above, it can be found that Web application, integration of various communication protocols, accommodation of value-added services, integration with enterprise application and adoption of Web Services are the trends of IB integration.

A. Information Model for Various Communication Protocols

Since BACnet, EIB objects and LonMark functional profiles are information models, and XML is a modeling language, these high level information models could

be expressed 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 had its own XML model, there would be 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. If BACnet, EIB objects, and LonMark functional profiles are methods of modeling information, what is needed is a unified information model to include these BAS protocols as well as other facility-related applications [17].

B. Value-added Services in IB Integration

Value-added services can be developed in IB integration system. The common method is offline calculation method nowadays. The value-added services software calculates according to the historical data from historical database, and then tells the control system the optimization parameters or FDD consequences. This method can tell whether facilities or entire building work on a good performance, and how to change the parameters for better performance. This method is an off-line method, cannot adjust the HVAC running parameters instantly. However, the working condition, such as weather, outside temperature, is changed continually, the calculated

parameters may not be best when the conditions change.

From the view point of implementation architecture, the normal traditional method is to embed the value-added services into the supervisory software. The value-added services are hardcoded in the supervisory software as one part of the latter. However, this method has disadvantages of inflexibility. It needs the software manufacturer to cooperate to recompile the source code when some new services need to be added or changed. The designers of optimization applications cannot add and modify their implementation methods flexibly as they wish.

C. Integration with Enterprise Application

The integration with enterprise application is data-sharing between BAS and enterprise applications which have two different meanings, batch data processing and real-time data processing. In the batch data processing way, the data, configuration, and report can be exported from BAS to enterprise application, vice versa. The export/import usually is made on the basis of scheduled tasks or personal operations. In the real-time data processing way, data exchange between BAS and enterprise applications can be made frequently. The information in one side can be transported to the other side instantly.

Nowadays, due to the lack of common standard interface, the communication and data-sharing between BAS and enterprise application mainly is based on batch data processing. In this way, one side cannot get information from the other side instantly.

D. Web Services in BA Integration

Although oBIX and BACnet/Web Services define the standard interface to provide building data to enterprise systems across the Internet or an Intranet, however, little information on design and implementation of integration based on Web Services has been published.

The currently-used integration software systems usually realize integration and interoperation of BASs in the LAN. The access method to building management system (BMS) on the Internet is provided by browsing web pages on the proxy web server computer as well. However, these solutions haven't truly achieved the integration based on the Internet. They integrate sub-systems based on LAN, only providing Web pages which end-user can use to access the BAS by web browser over the Internet. What they realized is an interactive interface to end-user, instead of mutual data communication among BASs. They cannot realize communication among BASs distributed on the Internet. It may be more suitable to call them "user web-access" or Web pages-enabled. They are not integration among BASs based on the Internet.

1.3 Aim and Objectives

The aim of this research project is to design and implement an Intelligent Building integration platform based on the latest middleware technologies, providing integration of various BA sub-systems and value-added services on the Internet. This aim is achieved by addressing the following objectives:

A). *Design bi-directional information transportation based on Web Services*

In the service model of Web Service, the normal control and monitoring functions for BAS are achieved by the client-request-server-reply communication process. The information transportation of Web Services is based on http or other Internet protocol. Web Server/Web Services are connectionless and stateless (HTTP is a stateless protocol). It is a problem how to breakthrough the restriction of Web Service to make Web Service as the bi-directional standard communication technology to transfer real-time and historical data on the Internet.

B). *Develop an extensible and scalable integration platform*

A middleware platform will be designed and implemented for the integration of intelligent building systems. As discussed before, this platform will adopt Web Services and object-oriented technologies to realize a unified integration model. The platform will have the features as listed as below:

- Based on XML/Web Services
- Flexible integration/interoperation among heterogeneous BAS systems
- Accommodate various communication interfaces
- Integration with enterprise applications on the Internet
- Unified data/object model
- Provide value-added services.

C). *Implement and Validate Software Platform in A Large Building*

This integration platform is implemented and validated in a large building – International Commercial Center (ICC). In the research of the HVAC system

optimization of the ICC project, a few optimization services will be implemented based on this middleware platform. In our group, some team members are engaged in FDD and data fusion research of HVAC system. FDD can diagnose whether a temperature sensor is bad or greatly value-shifted by analyzing data from other sources or sensors [15]. Data fusion is the fully automated method of merging diverse data into a single, coherent representation of a tactical, operational or strategic situation. For example, cooling load of a building can be calculated by flow rate and temperature difference of chilled water, however the measurement of the flow rate may have great gap with the true value. The cooling load can be adjusted by data fusion method with the power consumption measurement [16]. All the HVAC optimization applications which belong to value-added services are realized based on the designed integration platform.

1.4 Organization of the Thesis

This thesis will introduce the design and implementation of the integration platform based on middleware technologies. Based on the trends and requirements of the industry and comparison of different middleware technologies reviewed, a unified integration unit (UIU) - IBmanager is presented. The IBmanager employs *standard communication protocol and distributed computing technologies, including object-oriented programming, data-subscription & event-driven technology, Web Services, XML driver technology, and value-added services plug-ins technology*, to realize data and services integration and interoperation among distributed BASs on

the Intranet/Internet. The implementation of this platform was made in the laboratory and is being deployed in an practical project – International Commercial Center (ICC) in Hong Kong. The evaluation of its performance is conducted primarily as well. This thesis is organized as follows.

Chapter 1 presents motivation of the research and conducts literature review of related study and application. Section 1.1 introduces the necessity of integrating BAS with enterprise system/value-added services and the current situation of diverse communication protocols in the market, concludes that developing a middleware platform to achieve integration among BASs on higher level, namely, management layer is very necessary. Section 1.2 reviews the evolution of BAS communication protocols and information technology, classifies the definition of system integration and compares various middleware technologies. Section 1.3 concludes the objectives to develop an integration platform based on Web Service technology and deploy a series of optimization services for HAVC applications based on it.

Chapter 2 introduces the issues and difficulties in IB integration and methodologies based on existing two possible technologies. Section 2.1 introduces technology difficulties in IB integration. Section 2.2 introduces the characteristic and its disadvantages of the OPC technology which was used as the main technology in the beginning of this study, the latest development of OPC is introduced as well. Section 2.3 introduces the details of Web Services technology and the related researches. Section 2.4 concludes Web Services technology is good choice for IB integration.

Chapter 3 presents the designed platform - IBmanager. Section 3.1 introduces system model of IBmanager and its chain-type deployment. Section 3.2 and Section 3.3 introduce the fundamental technologies in the IBmanager, bi-directional communication model based on Web Service and heterogeneous databases integration by Web Service.

Chapter 4 presents the object model of the IBmanager, including Driver Objects, Data Point Object, Database Agent Objects, Alarm/Event Objects, Interoperation Object, and Session Object. The corresponding functions realized by these objects are introduced as well.

Chapter 5 introduces some implementation issues of the designed platform, including its distributed XML Driver model, Human Machine Interface (HMI) based on Web, concurrent operation, redundancy and fault-tolerant, value-added services implementation.

Chapter 6 introduces test facilities and integration methods in the Intelligent Building Lab of Dept. Building Service Engineering (BSE), The Hong Kong Polytechnic University, together with communication performance test method.

Chapter 7 presents performance analysis and evaluation of the designed platform. The analysis of communication latency, integration and interoperation test, the roundtrip time test and application load test are conducted and introduced.

Chapter 8 introduces practical use of IBmanager platform in the International Commercial Center (ICC) project. The introduction of the ICC project, the deployment architecture, the functions realized are presented in details.

Chapter 9 presents conclusions of the integration platform and discussions about future work. Load-balancing, mobile application, security of Web Service, public services are some issues to be addressed in the future work.

CHAPTER 2 SYSTEM DESCRIPTION AND METHODOLOGY

2.1 Difficulties of System Integration

2.1.1 Incompatible Field Bus Protocols

There were dozens or more networks and communication protocols in use throughout the building controls industry. Multiple DDC Manufacturers with multiple generations of products contributed a proliferation of system networks based on countless proprietary protocols, commonly referred to today as "legacy systems". System users had long voiced concerns about the complexity of DDC system management and expansion. This was due to the inability of systems to share data or communication networks. The industry became aware of the role that communication played in the long-term success of DDC systems and the importance of protocols and networking. The incompatible protocols frustrate field engineers greatly when integration and interoperation is necessary. The same challenges remain today, but with a tremendous simplification, there are only several standards or common protocols shared the majority market rather than dozens protocols before [52].

The situation is getting worst when coming to broader integration. The other systems, including access system, security system, video surveillance, fire safety, and enterprise applications, introduce more communication protocols to join the protocols family to be integrated.

2.1.2 Disparate Data Models

Besides the disparate communication protocols, the data definition models from different protocols or products may be different. Let's elaborate it by BACnet and LonWorks as examples. One of the main and enduring values of BACnet is its data definition model, i.e., the way BACnet provides a comprehensive representation of the functionality of building automation and control equipment (its object model). LonWorks also has its data model on building automation, although it was not designed specifically for building systems [51]. BACnet and LonWorks have totally different data model.

Web Services provides a standard way to locate and gather data. But if every computer application has a different data model, the data may have limited use only. This has been proved to be a problem in some of the early Internet based business-to-business transactions that predate Web Services. XML provided a universal way to format the data, but if the data structures used by the two systems were fundamentally different there was still a lot of hand coding required. One suggestion has been to develop "vertical standardization," that is, standard data models within each industry. A data model that works in the Electric Utilities industry would probably not fit the needs of the Real Estate industry, but standardization within the utilities industry would make it much easier to develop a Web Service that would work with multiple utility companies [55].

2.1.3 Various Distributed Technologies

There are a series of middleware platform technologies. Different BAS manufacturers may have different favorites. DCOM, OPC, RMI, CORBA, etc., can be

found in the BAS applications. These middleware platforms provide great implementation vehicles for services, and have different characteristics, advantages and disadvantages. How to make them work coordinately is a challenging task.

2.1.4 Heterogeneous Database Management System

There are different database management systems in the BAS applications, including Access, SQL Server [93], MySQL [94], even proprietary database. The differences among the database management systems include [54]:

Technical heterogeneity: different file formats, access protocols, query languages etc., often called syntactic heterogeneity from the point of view of data;

Data model heterogeneity: different ways of representing and storing the same data, it also is referred as schematic heterogeneity;

Semantic heterogeneity: data across constituent databases may be related but different.

The integration software needs to retrieve information from a combination of databases which have been constructed in different ways, i.e., heterogeneous databases. The common problems of heterogeneous database access occur where databases are constructed that share some elements, yet are different in any of several ways, such as [72]:

- The databases follow different commonly used models – e.g. object-oriented, relational, and hierarchical.
- Missing or conflicting data – where the same data are entered in two

databases that may conflict. Recording-time and reliability metadata will be required to choose between alternatives.

- The structure of the models could be different.
- Different levels of abstraction may be used to model the same entities – e.g. “price” may be defined as including local taxes, or it may be left implicit as to whether they are included.
- The defined scope of concepts in models may vary – e.g. “price” may include or exclude service charges or local taxes.
- The representational structure of objects and properties in the model may vary.
- The role in the model may differ for the same object to convey its role in different processes – e.g. the same person may be entered in a database as both a “customer” and a “supplier” for different products.
- Precision – e.g. a raw score or a rounded score may be stored.
- Format differences: data type – e.g. a part number as an integer or alphanumeric string.
- Units or measurement scales may vary – e.g. the units of currency may be Euro or US dollars.
- The name used for the same concept is different – homonym, e.g. “motor”, “car”, “auto” for the same motor vehicle.
- Languages may vary – cross-language homonym e.g. the natural language from which the element label is derived may be different.

2.2 Integration Based on OPC Technologies

At the beginning of this research, OPC is selected as the main technologies to achieve the integration platform. After deep study and development for some time, Web Services is adopted instead of OPC as the main communication technology. Even so, OPC is investigated as well for comparison.

2.2.1 Investigation on OPC

OPC stands for "OLE for Process Control", it is a communication standard based on OLE/COM technology. It brings the same benefits to industrial hardware and software that standard printer drivers brought to word processing [55]. Based on Microsoft's OLE, COM (Component Object Model) and DCOM (Distributed Object Model) technologies, OPC consists of standard interfaces, properties and methods for use in process control and manufacturing-automation applications.

The OLE/COM/DCOM technologies define how individual standard components can interact and share data. OPC provides an interfacing standard for factory automation where every system and every communication driver can freely connect and communicate. Having such a standard, the communication and interactions between different applications, from the plant to the MIS (Management Information System), becomes easier with truly open enterprise communication as Figure 2.1 [55].

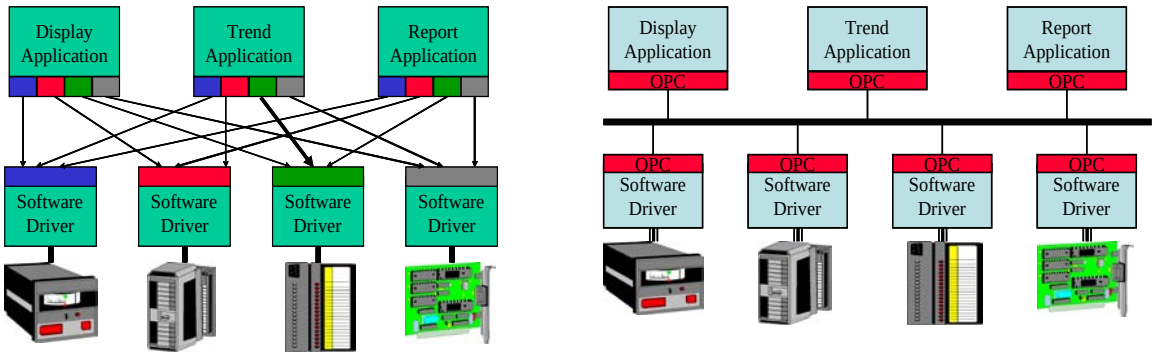


Figure 2.1 Architecture compare of integration based on traditional driver and OPC

The application of the OPC standard interface makes possible interoperability between various systems in the BAS management level as Figure 2.2. Traditionally, each software or application developer was required to write a custom interface, or server/driver, to exchange data with hardware field devices. OPC eliminates this requirement by defining a common interface that permits this work to be done once, and then easily reused by HMI (Human Machine Interface), SCADA (Supervisor Control and Data Acquisition), control and custom applications [56].

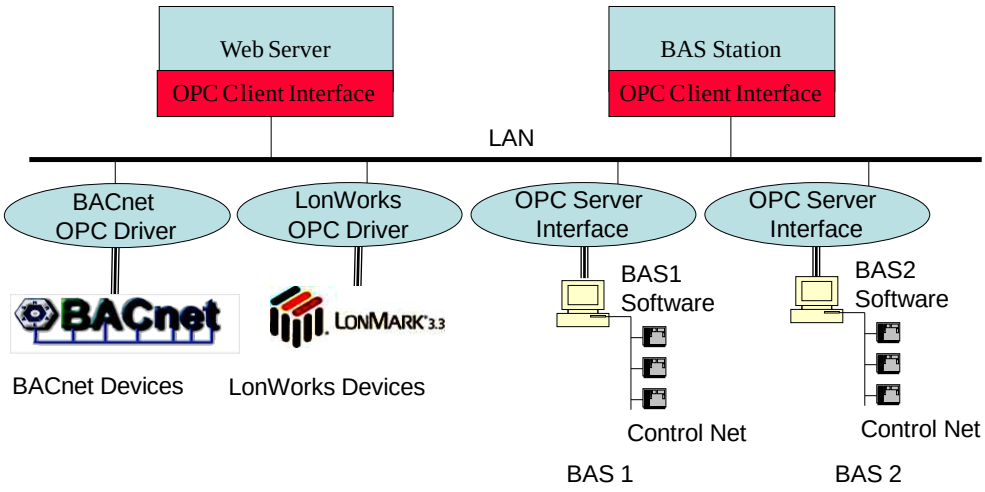


Figure 2.2 BAS integration via OPC

In Figure 2.2, BAS 1 and BAS 2 software systems have provided the OPC server interface, so BAS station software and Web applications can communicate with them via the OPC interface. Another situation is that a standalone OPC server is provided with a device. In this case it is easier for the devices being connected directly via OPC server. Both the standalone OPC server and BAS software with OPC interface provide data to the BAS Station, they can be distributed on different computers other than the BAS Station. New BA system/devices with OPC interface/driver can be added with little difficulty for the unified interface.

2.2.2 Disadvantages of OPC

However, this implementation with OPC still has problems as discussed as below.

(i) No enough complex data structures definition

OPC is not a new standard bus or a universal protocol, but it is an interface definition defined by different companies from industrial automation and Microsoft. The complex data structures definition are not yet fully defined in OPC, leaving them dependent on the application. From the view point of a driver, it is still vendor specific.

Earlier OPC specifications failed to provide a single coherent data model - e.g., the Data Access item hierarchy was totally disjoint from that offered by Alarms & Events [57].

(ii) Not suitable on Internet

OPC uses COM/DCOM as the core technology for the software interface. Therefore, when an OPC client on a computer connects to an OPC server located on another computer, the DCOM security must be configured correctly. Many installers experienced this requirement as a problem. As a result, DCOM security is often disabled, leading to severe security risks. Obviously, it gets even more risky when using an OPC server over the Internet [58]. When you try to use DCOM over the Internet, you will likely be thwarted by DCOM's tight coupling with Microsoft Windows NT security and its use of dynamically allocated TCP/IP ports (not typically allowed through corporate firewalls) [59]. Due to these problems, it is not practical to use it over the Internet.

XML technology has been used to fill these gaps in a new development of OPC technology. The latest progress is the new OPC XML-DA (Data Access) standard. In this new standard, OPC allows manufacturers to process data which can be accessed via the Internet. In this case, the OPC server is configured as a Web Service [60].

(iii) Windows Platform Specific

Microsoft designed COM/DCOM as a modern, object-based RPC (remote Procedure Call) mechanism to facilitate cross-component, cross-process and cross-machine communications used in the Microsoft Windows environment. Although Microsoft has opened the door to non-Microsoft implementations,

COM/DCOM never really caught on outside of Microsoft, and is not available on most non-Microsoft platforms [59].

(iv) Difficult to Connect with Enterprise Application

Manufacturing enterprise applications like MRP (Material Requirement Planning) need real-time plant floor data that is often available via servers that implement the OPC-COM interfaces. The problem is that most of these higher-level applications do not implement the OPC-COM interfaces necessary to talk to OPC-COM servers. Many cases these applications are on non-Microsoft platforms, making it all but impossible to communicate via COM [59].

2.2.3 Latest Development of OPC Technologies

In the duration of this study, the OPC Foundation is struggling to re-design the OPC architecture to give OPC new life.

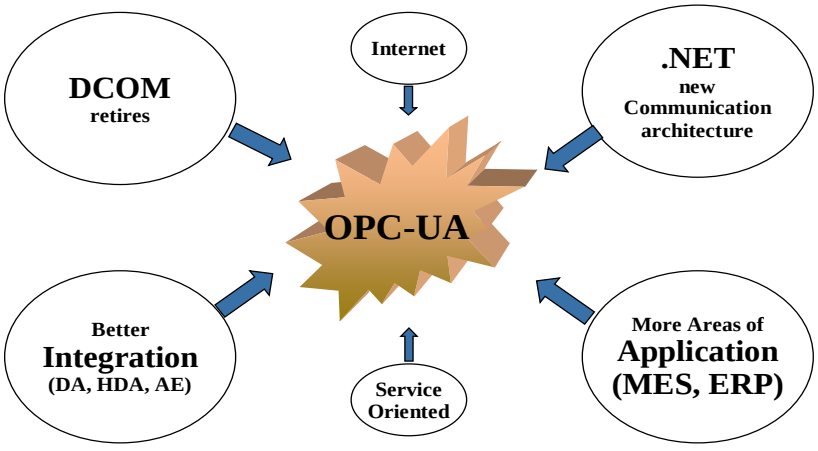


Figure 2.3 Latest developments of OPC technologies

The next generation of OPC - OPC Unified Architecture (UA) is being developed,

whose first parts of specification were released in June 2006. OPC Unified Architecture (UA) adopts Web Service technologies and unifies functionality across the existing OPC specifications as Figure 2.3 [57]. The UA initiative offers a single coherent data model, and uses Web Services for primary transport [61].

2.3 Integration Based on Web Service Technologies

Web Services is the fundamental building block in the move to distributed computing on the Internet. Open standards and the focus on communication and collaboration among people and applications have created an environment where Web Services are becoming the platform for application integration. Applications are constructed using multiple Web Services from various sources that work together regardless of where they reside or how they were implemented.

Web Services are built on XML, SOAP, WSDL and UDDI specifications. These constitute a set of baseline specifications that provide the foundation for application integration and aggregation. From these baseline specifications, companies are building practical solutions and getting real value from them.

2.3.1 The Core Technologies of Web Services

(1) XML

XML is a derivative of SGML, the Standard Generalized Markup Language, which is a standard, vendor-independent and platform-independent language way to

represent and store data. HTML, the language used to create the web pages that you see in your browser, is also a language that was originally created as a part of SGML. While being well suit to displaying information, HTML does not provide the structure necessary to organize and exchange data. XML allows data to be stored in a human readable file format that can be viewed using a web browser or even a word processor. Other tools available on the market, and many free or shareware applications, allow XML to be viewed and edited in a hierarchical way, much like Microsoft Explorer allows the files and directories on your hard drive to be viewed in an intuitive “tree” structure. The reason the data can be represented in this familiar way is that XML uses tags, much like HTML data tags, to record the relationships between the data elements. When an XML data file is read, it is easy to see that the device is called “Controller” and it contains objects such as points, messages, and alarms. Using HTML, it would be impossible to represent those points, messages, and alarms are part of the “Controller”, but instead the data would appear to be a simple list [62].

Here is an example of what's contained in an XML file that might be used to configure a controller:

```
<Controller>
        <Point1>
            <name>Outside Air Temperature</name>
              <conversion-formula>Degrees F</conversion-formula>
            <location>Near Door 7</location>
        </Point1>
```

```
        <Point2>
            <name>Space Temperature</name>
            <conversion-formula>Degrees F</conversion-formula>
            <location>Ground Floor</location>
        </Point2>
    </Controller>
```

Although XML allows data to be represented, stored and recalled, it does not address the issue of what kinds of data should be used to represent a particular physical or software object. To solve this problem, the XML community has adopted the idea of a schema. A schema is simply a definition that says if you wish to represent something called a "Point" object, to use the previous XML example, it should contain the properties called name, conversion-formula, and location. While XML is the language that allows the data exchange, the schema is the agreement as to what types of data will be necessary as part of the exchange.

(2) SOAP

SOAP is a protocol for exchanging XML-based messages over computer networks, normally using HTTP/HTTPS. SOAP forms the foundation layer of the Web Services protocol stack providing a basic messaging framework upon which abstract layers can be built. There are several different types of messaging patterns in SOAP, but by far the most common is the remote Procedure Call (RPC) pattern, in which one network node (the client) sends a request message to another node (the server) and the server immediately sends a response message to the client [64].

The SOAP specification defines what an HTTP message containing a SOAP message must look like. This HTTP binding is important because HTTP is supported by almost all modern operating systems (and several not so modern operating systems). The HTTP binding is optional, but almost all SOAP implementations support it because it is the only standardized protocol. For this reason, there's a common misconception that SOAP required HTTP. Some implementations support MSMQ, MQ Series, SMTP, or TCP/IP transports [63].

The SOAP specification defines the structure of an XML document that can be used to exchange data between two applications. It defines a way to represent programming language specific data types in XML, although it is not required. The SOAP specification defines a way to use SOAP to request-response RPC style messaging, but does prevent you from using other styles of messaging. It also defines a way to exchange SOAP messages over an HTTP transport, but doesn't limit you to using that transport. This flexibility means that SOAP is widely applicable to a large number of communications requirements [63].

(3) WSDL

The Web Services Description Language (WSDL, pronounced 'wiz-dəl' or spelled out, 'W-S-D-L') is an XML-based language that provides a model for describing Web Services. For our purposes, we can say that a WSDL file is an XML document that describes a set of SOAP messages, and how the messages are exchanged. In other words, WSDL is to SOAP as the interface definition language (IDL) is to CORBA or

COM. Since WSDL is XML, it is readable and editable, but in most cases, it is generated and consumed by software [63].

WSDL is often used in combination with SOAP and XML Schema to provide Web Services over the Internet. A client program connecting to a Web Service can read the WSDL to determine what functions are available on the server. Any special data types used are embedded in the WSDL file in the form of XML Schema. The client can then use SOAP to actually call one of the functions listed in the WSDL. WSDL can make it much easier when exposing SOAP services for others to call or consuming SOAP services.

(4) UDDI Services

UDDI Services, a dynamic and flexible infrastructure for Web Services. This standards-based solution enables companies to run their own UDDI (Universal Description, Discovery, and Integration) directory for intranet or extranet use, making it easier to discover Web Services and other programmatic resources. Developers can easily and quickly find and reuse the Web Services available within the organization. IT administrators can catalog and manage the programmable resources on their network. Enterprise UDDI Services also helps companies build and deploy smarter, more reliable applications [65].

UDDI Services provides easy discovery of Web Services and other programmatic resources inside an organization. Two common scenarios for UDDI Services inside an

organization are Developer Reuse and Dynamic Application Configuration:

- Developer Reuse. At design time, developers search UDDI Services for Web Services and other programmatic resources to reuse in building new applications. UDDI Services exposes all of the information needed to invoke a service, making it easy for the developer to integrate the service into an application.
- Dynamic Application Configuration. At run time, an application queries UDDI Services to discover the current binding information for the services it needs, and then connects directly to those services. An example of this is a stock broker application that queries UDDI Services first thing in the morning to get configuration information for the different services that are part of the application, such as a stock ticker, customer service updates, or settlement services [65].

2.3.2 Integration Based on of Web Services Technologies

This research aims to design and implement a middleware platform for IB integration based on Web Services technologies. As discussed above, oBIX and BACnet/WS mainly provide definitions of data and service model by Web Services, not provide a total solution for implementation or application.

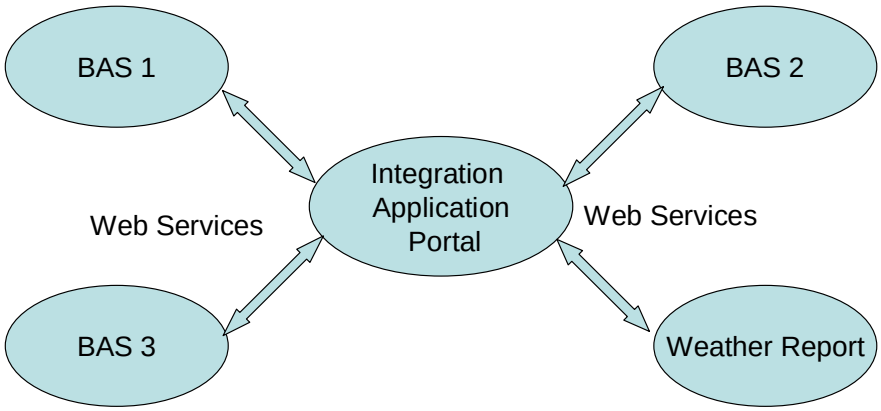


Figure 2.4 Integration of BAS systems using Web Services

Using the Web Service technology, BAS systems from different vendors, even on different platforms can be integrated easily. As shown in Figure 2.4, the Portal application can access different BASs via Web Services, the non-BAS systems can be easily integrated as well. 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 [17]. However, since the SOAP request/response is enveloped in XML format, it will be too complex to be used in the communication of field level control in some situation and will increase the need to the processor power and additional response time. Therefore, it is not suitable for field level up to now because of much traffic overhead [6]. Meanwhile, the traditional technologies have been broadly used in this industry. So the process of monitoring and controlling the field devices or BAS networks still needs to resort to field bus communication protocols or traditional middleware technologies.

Upon to the beginning of this study, practical research about the application of Web Service on IB is not sufficient, especially how Web Services technology coordinates with field level devices and traditional middleware technologies and how to accommodate value-added services is not addressed in detail.

2.4 Conclusive Remarks

The traditional OPC technology has been used in the automation industry for many years with broad popularity. However, with the development of IT and broad application of Internet technologies, it is not suitable as a fundamental technology of intelligent building integration platform. OPC Foundation has addressed this problem and presented a new OPC Unified Architecture, which is based on Web Services. Almost at the same time of this progress, we gave up traditional OPC and resort to Web Services as well.

In the jungle of distributed technologies, since it is text-coded and based on standard protocols, Web Services is a good choice to integrate BAS systems from different vendors compliant with different protocols, even on different platforms. The chapters behind will elaborate how the designed middleware platform communicates with field level devices and traditional middleware by Web Services, and how to provide a unified data points and services for upper-layer applications.

CHAPTER 3 OVERALL SYSTEM MODEL

3.1 Overall System Design

3.1.1 Principles of System Design

The integration platform is designed to meet the principles as below:

- *Object-Oriented:* BMS functions are capsulated into objects. This will benefit the BMS with better scalability and extensibility;
- *Communicate by event or message between components:* The components are loosely-coupled by communicating with events or messages. This will increase the system stability and robustness;
- *Data-subscription & event-driven:* BMS needs to realize many automation functions (alarming, scheduling, data recording). Some of them are optional to be realized and some will be implemented only to specified data points. Polling every data point will be a great waste of time. The data-subscription & event-driven are adopted to reduce this overhead. Only the data points being subscribed with specific functions will process these functions;
- *Scalability and Extensibility:* The platform can be suitable for different-sized scales applications. The entire platform is comprised of a series of modules, which can be loaded dynamically according to the requirements of specific project. The new BAS product/system can be added just by building the driver according to the driver specification;

- *Flexible deployment:* System can work standalone as a full-functional BMS, or work together with other applications or other instances of this platform;
- *Self-contained, autonomous unit:* The different deployed instances of the designed platform can cooperate together, meanwhile every deployed instance is a full-functional unit which can achieve the all functions of BMS standalone by itself;
- *Standard communication technologies:* The standard communication technologies, such as BACnet, OPC, Web Services must be accommodated to keep this platform extensible;
- *Internet-friendly:* Internet has greatly influence people's life and work. It is a must-have capability to work over the Internet for this platform;
- *Integration with enterprise application:* It must exchange data with enterprise applications.

3.1.2 Design of Integration Platform

This section presents the system model of the integration platform designed using the most updated technologies, including object-oriented, data-subscription & event-driven, XML and Web Services to realize integration and interoperation of intelligent building system. This middleware platform, namely "IBmanager", is designed as "Unified Integration Unit (UIU)", which aims to realize the integration of information and services among BASs in the LAN or over the Internet by M2M (Machine to Machine) communication, not just providing web pages for users to access which exists popularly nowadays.

The IBmanager is designed to integrate various sub-systems/devices to share data and information. It can work as an autonomous, self-contained unit as Figure 3.1. Its core part is the **Application Server (App. Server)**. Together with the front end technologies (Human Machine Interface), distributed driver technologies, database agent, it forms a full-fledged BMS (Building Management System).

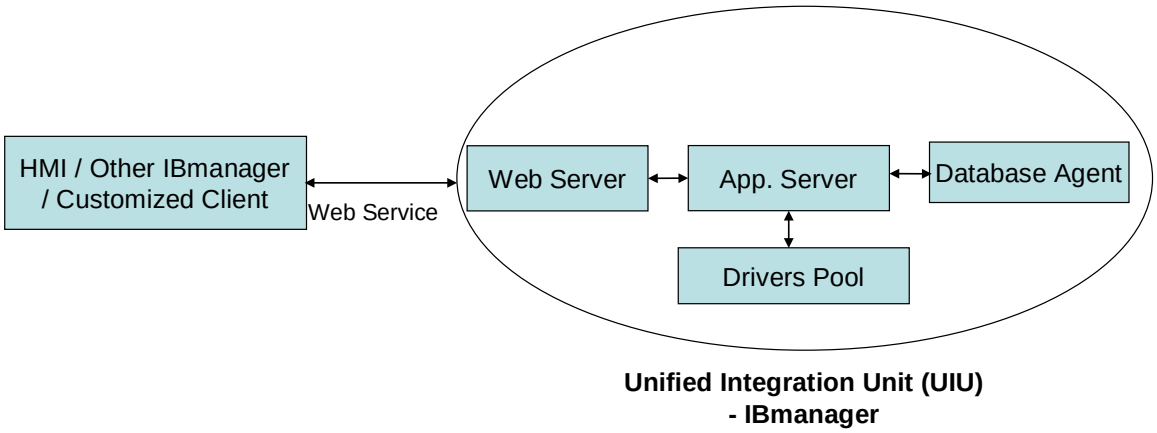


Figure 3.1 Architecture of Unified Integration Unit (IBmanager)

In the IBmanager, the **Drivers Pool** is responsible to communicate with other BAS/BMS software or control networks directly. The standard Web Services communication driver has been made as default driver in the implementation of the Drivers Pool, meanwhile other kinds of communication protocols can be developed for the IBmanager according to the driver interface specification. **Application Server** (App. Server) is the core part of IBmanager, the operation logic and BMS high-end functions are realized in it. **Web Server** here acts as http parser for Web Services invocation besides its original functions. **Database Agent** is in charge of accessing local database and remote database. There are two kinds of interfaces to database, one is to access local database via ODBC, SQL technologies, and another is to access remote database via Web Services. The database caching, query decomposition and

response composition are included in Database Agent, whose details will be presented later. The “HMI/Other IBmanager/Customized Client” can be the Human Machine Interface for user to access, another IBmanager installation or customized clients. The communication between HMI (or other IBmanager installations, customized client) and the IBmanager employs a high-level M2M interface using Web Service.

The IBmanager have two kinds of typical application cases. In the first case it acts as a standalone application in which it has been deployed with full-functional HMI Web pages, user can access this system by popularly-used web browser. In the second case it just acts as a data source to be integrated into other IBmanager installations or customized client applications by Web Services interface. In this case, HMI for the IBmanager is not necessary. The main difference between these two cases is whether HMI is deployed, the main parts of the IBmanager is identical. These two typical applications are not exclusive. They can exist on the same time as Figure 3.2.

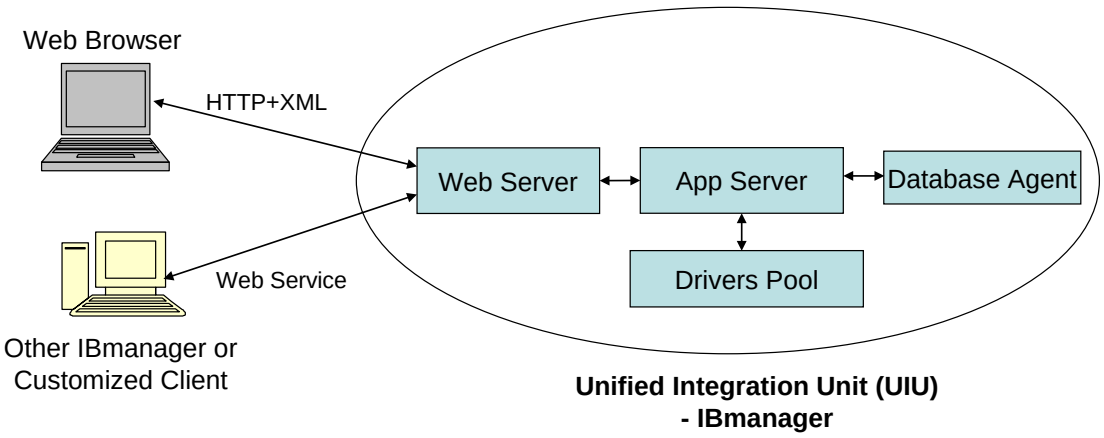


Figure 3.2 Two typical application cases of the IBmanager

3.1.3 Three-Tier Model

The IBmanager meets the three-tier model as Figure 3.3. **Driver layer** is

responsible to communicate with different products with various protocols from field device/network or high-level BAS software. A driver configuration tool is provided to configure the drivers and data points for the IBmanager, including the functions of add/edit/delete drivers and data points. This is useful to load data points and drivers dynamically. In the driver layer, the physical data points can be given a human-friendly name for convenient configuration and easy access by upper-layer.

Application layer accomplishes the building management functions, operation logic and the access to local and remote database. The access to local and remote database is processed by a database agent. A logic configuration is used to configure the operation logic of application layer.

Presentation layer presents the human interface to users or communication interface to other IBmanager installations or customized client applications. For human access, the web HMI is presented in this design. Display authoring tool is used to build the display pages for HMI. Users can use the popularly-used web authoring software as well, such as FrontPage, Dreamweaver.

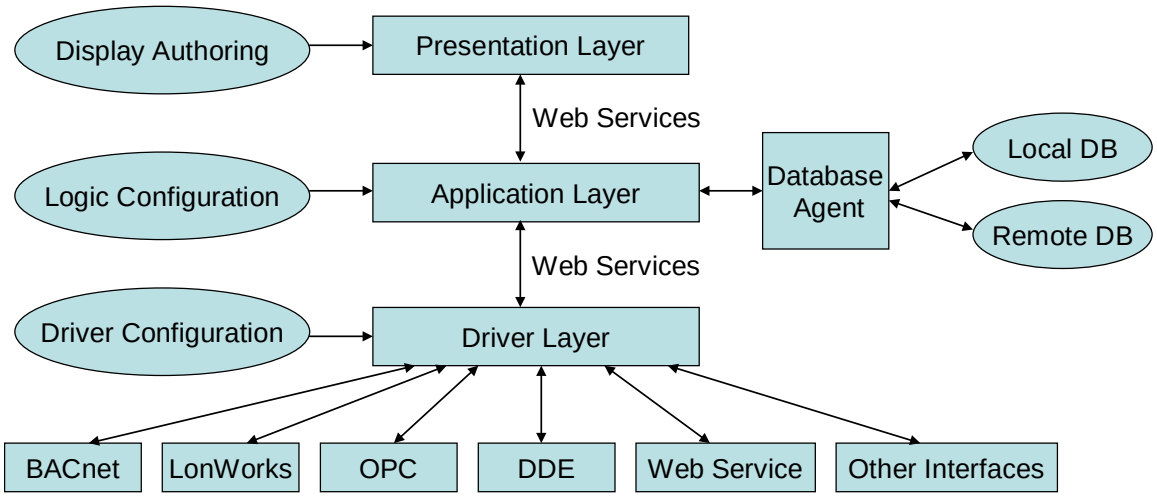


Figure 3.3 Three-tier architecture

3.1.4 Network Topology

Figure 3.4 shows the networks topology of this integration and management platform. In the bottom right corner of this figure, the IBmanager-2 doesn't provide web pages to be accessed by users, just provides public Web Services interfaces to be accessed by other applications or other IBmanager installation. It is a data source for integration of greater scope applications.

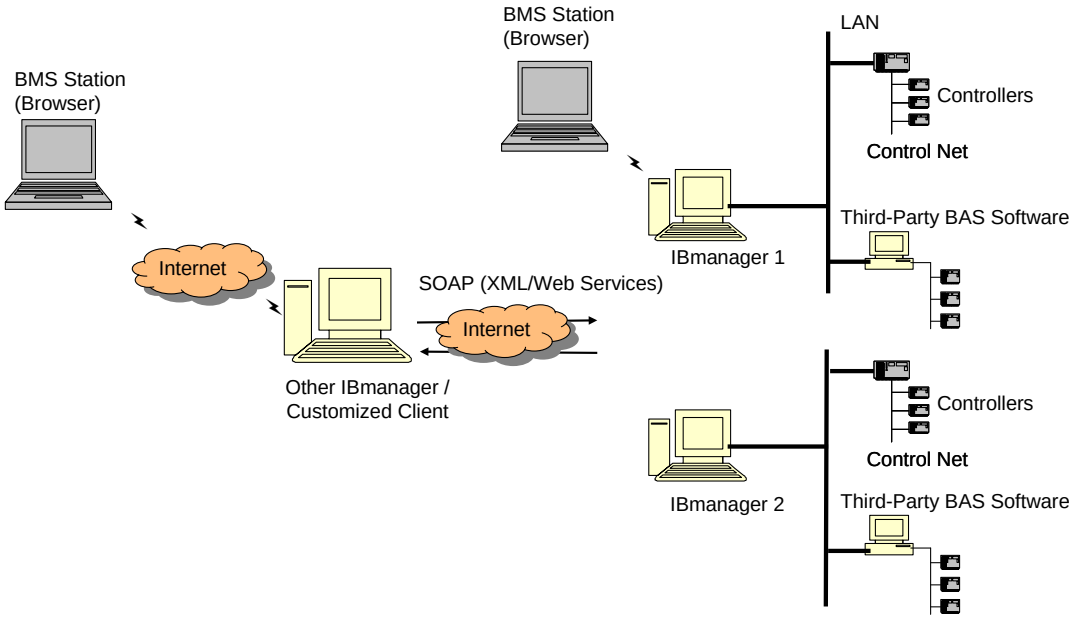


Figure 3.4 Network topology of BAS integration

In the top right corner of this figure, the IBmanager-1 is full-functional BAS software which provides building management functions for the corresponding BA system/network. It provides web pages to be accessed by users and can work as a standalone building management system. At the same time it provides public Web Services interfaces to be accessed by other applications or other IBmanager installations as well.

The main difference of these two application cases is on their direct client and whether the IBmanager is equipped with HMI. The direct client in the former case is another IBmanager installation or other application, it is M2M communication with no need of HMI, the direct client in the latter case is HMI for human access. In both cases, users can easily develop their own applications to monitor and control the BAS system by the Web Services interface. Facility Management company can manage, monitor and control their managed BASs by browsing the web pages, or make some value-added services including decision analysis or data mining.

3.2 Chain-type Deployment of IBmanager Installations

Although the system supports any kind of driver as long as the driver is compliant to the driver development specification, Web Service driver is natively supported. With the Web Service support, the Drivers Pool of one IBmanager installation can be consumer/client of the Web Services of another IBmanager installation. In this case, the former IBmanager acts as consumer/client of the latter IBmanager. Thus the IBmanager installations are deployed as chain-type installations. From the view point of topology, the IBmanager installations can be deployed as chain-type horizontally or vertically.

3.2.1 Vertical Chain-type Deployment

Figure 3.5 is the example of vertical chain-type deployment. In vertical chain-type deployment, IBmanager installations are deployed in different levels, from

control network, building network to enterprise network. The directions drawn between IBmanager installations are the integration directions instead of communication directions. The deployed installation at lower level works as data source of the deployed installation at higher level. The IBmanager is designed to be suitable from small-scope to large-scope applications as a unified integration unit. This embodies the scalability of the IBmanager.

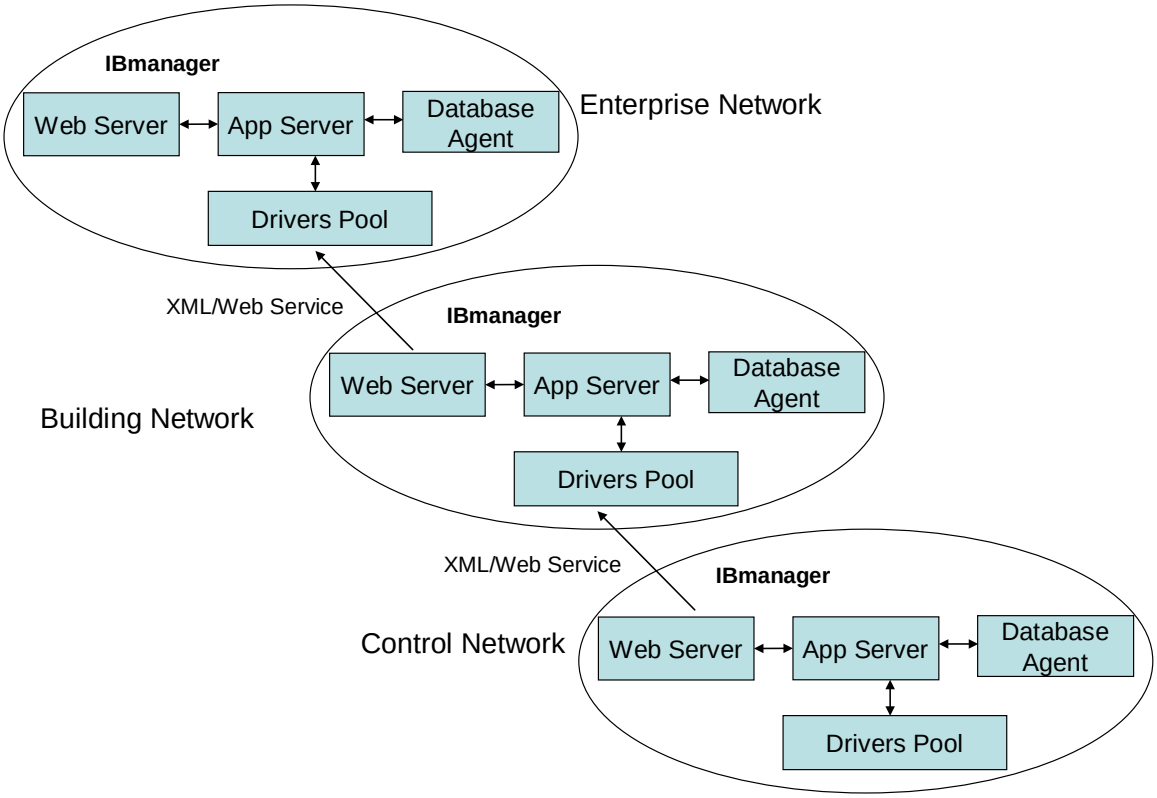


Figure 3.5 Vertical chain-type deployments of IBmanager Installations

Although the IBmanager can be deployed in various levels with the same architecture, it has the flexibility to adjust the configuration according to the characteristic of the deployed level. The IBmanager deployed in higher level integration can extract the data points which it is interested in from the IBmanager

installations deployed in lower level, ignoring the data points what it is not interested in. Even in the data points extracted, the IBmanager installations deployed in high level can just care about part information of these data points. The IBmanager installations deployed in high level can create their own virtual data points and calculate the values of these virtual data points based on the IBmanager installations deployed in low level. These virtual data points usually are made and calculated for decision-making or data fusion from a great volume of data points.

3.2.2 Horizontal Chain-type Deployment

Figure 3.6 is the example of horizontal chain-type deployment. In this case, the IBmanager installations are deployed in the same level. The data sharing and interoperation is achieved across the IBmanager installations. Every IBmanager installation can integrate other IBmanager installations as its sub-systems by Web Service driver. In this deployment case, IBmanager-1 can integrate the information of IBmanager-2, vice versa. These two IBmanager installations work in parallel.

The data points managed in one IBmanager installation can be accessed by the client of another IBmanager installation. This is conducted by two methods. One method is direct access, that is, the client accesses all the IBmanager installations directly by Web Services. The other method is indirect access, the original data points in the former IBmanager installation have been integrated into the latter IBmanager installation as its own data points. Thus the client accesses the data points of the former IBmanager installation through the latter IBmanager installation. In the latter

method, information from different IBmanager installations can be integrated into one IBmanager installation, so the client of one IBmanager installation can access information of all other IBmanager installations by this way. Similarly, users can develop a Web Services portal to access all the IBmanager installations with the integration of Web Service methods provided various IBmanager installations. This integration is not a simple integration of web pages, but also the integration of data and services based on M2M communication.

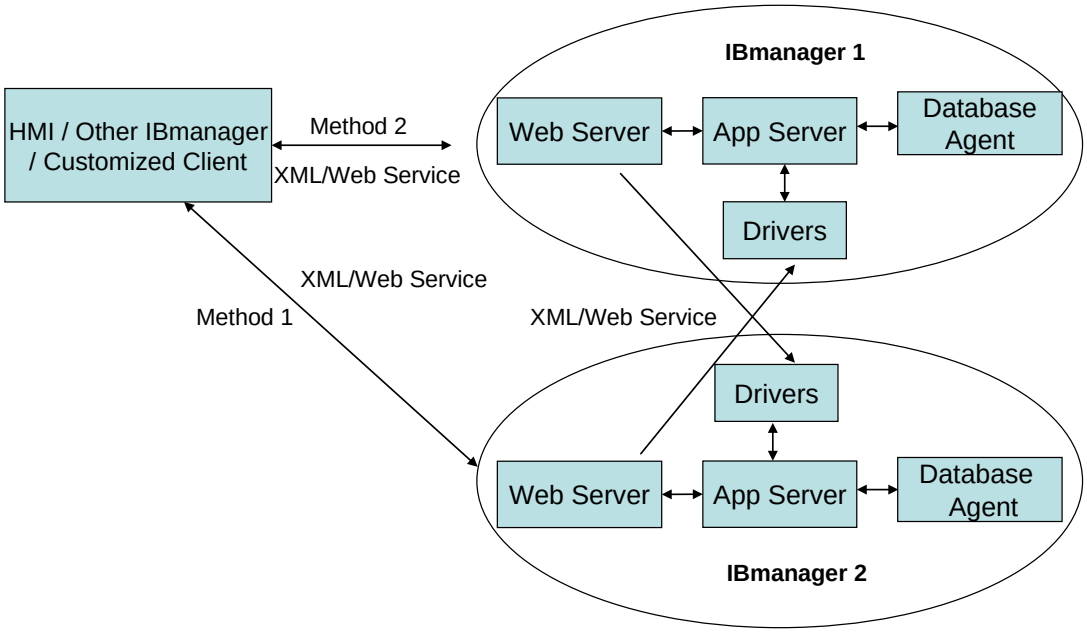


Figure 3.6 Horizontal chain-type deployments of the IBmanager installations

3.3 Bi-Directional Communication Model Based on Web Service

3.3.1 XML/HTTP Request/Response Model

HTTP messages consist of requests from client to server and responses from server to client.

HTTP-message = Request | Response ; HTTP/1.1 messages

Request and Response messages use the generic message format of RFC 822 for transferring entities (the payload of the message). Both types of message consist of a start-line, zero or more header fields (also known as "headers"), an empty line (i.e., a line with nothing preceding the CRLF) indicating the end of the header fields, and possibly a message-body [69].

generic-message = start-line

*(message-header CRLF)

CRLF

[message-body]

start-line = Request-Line | Status-Line

The XML encode message is transferred as http payload. The example of XML message over http [70]:

Send a XML request:

<TransferRequest>

<Method> SynReadPoint </ Method >

<Point> Outside Temp </ Point >

</TransferRequest>

Receive a XML Response:

<TransferResponse>

<Method> SynReadPointAck </ Method >

<Point> Outside Temp </ Point >

<Value> 23.6 </ Value >

</TransferResponse>

3.3.2 The Restriction of Web Service

In the service model of Web Service, the normal control and monitoring functions for BAS are achieved by the client-request-server-reply communication process. For example, in the IBmanager architecture, clients (i.e., HMI, another IBmanager installation in chain-type deployment or customized clients with Web Service consumer interface) submit requests to the services provided by the IBmanager. The App. Server of the IBmanager executes specific services upon requests and returns data to the clients.

However, some BAS functions need a contrary communication process at the same time, that is, the server sends data to its clients automatically. This occurs in case of the transportation of notifications, such as Alarm/Event (A/E) and Change of Value (COV). When such alarms occur, client should be notified immediately since the alarm may cause serious consequences. In these cases, alarm/event and COV information are necessary to send from server-side (the IBmanager) to client (HMI, another IBmanager installation or customized clients) as Figure 3.7.

In the other technologies, such as DCOM world (and also in other environments) there are such things as callbacks, i.e., the direction of interaction changes. When this

occurs, the server invokes a method at the client. In OPC DCOM DA this feature is used to deliver data automatically from the server to the client if the data has changed [71]. However, the communication interfaces which the IBmanager releases are Web Services. This “callback” feature is not available with Web Services. The information transportation of Web Services is based on http or other Internet protocol. Web Server/Web Services are connectionless and stateless (since HTTP is a stateless protocol). Each server call is self-contained and independent of previous calls. The server initiated callbacks are not possible, the client always has to request the server for data. This kind of communication process belongs to so-called “pull” approach in which only clients can initiate the communication process to get information from servers and servers cannot initialize sending messages to clients. So how to realize the alarm/event and COV based on Web Services communication is a problem to be solved.

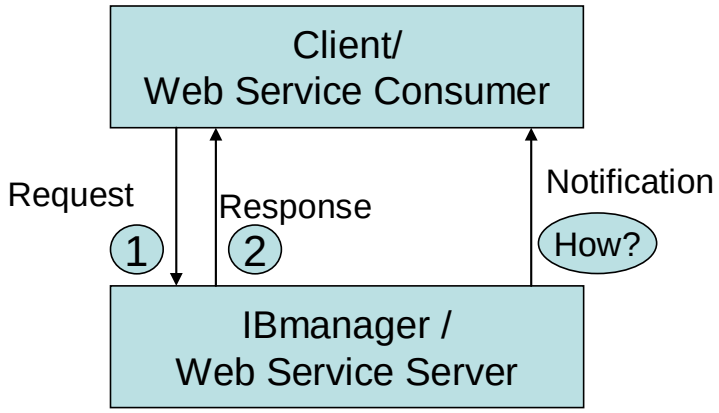


Figure 3.7 Communication method of Web Service

3.3.3 Reverse Message Transportation Methods

In order to realize sending message from server to client automatically, we need analyze the different deployment cases of the IBmanager, and make corresponding solutions. In one case, multiple IBmanager installations are chain-type deployed, the notifications and bi-directional communication needs to be realized between the IBmanager installations. Every end of communication is IBmanager installation, which is Web server since Web Service server must be Web server (http server) at the same time. Every end can be a client to initiate request and a server to make response at the same time. Every end can send out COV and alarm/event messages as a Web Service client at the same time when necessary. They can make bi-directional communication by “Peer to Peer” mode as Figure 3.8.

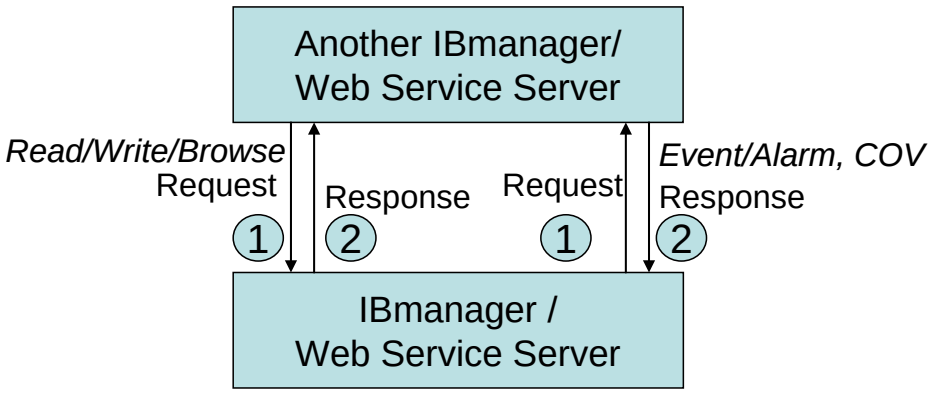


Figure 3.8 Application case of chain-type IBmanager installation

Another case is the communication between the IBmanager and its client which is installed in a computer without Web server, this client may be HMI or other applications discussed above except the IBmanager. The host of the client application

usually has not construct an http server environment to be ready to receive the http request, so the IBmanager cannot send “callback” messages, including COV, alarm/event messages to the client. To provide the way in which the IBmanager sends out notifications to client by the Web Services, two methods are designed for this aim.

Method 1: Peer to Peer Technology

One method defined as “Peer to Peer” method can be used to realize the “push” communication from the IBmanager to its client. Using this method, the client is also configured as a Web Service server, but it only provides the minimal set of Web Services to deal with A/E and COV messages. Therefore, it actually is a mini Web Services server. Its operation process is illustrated in Figure 3.9. Firstly, the client subscribes the A/Es and COV events in the IBmanager, the IBmanager will maintain a subscriber table, which contains the subscribers’ IP and events subscribed. When an event occurred, the IBmanager needs to send notifications to the clients by sending a Web Service request to the client. The client will analyze the event type, deal with it, and then return an ACK (acknowledgement) Web Service message to the IBmanager. Thus, a “Peer to Peer” communication is realized [6]. In this method, it has no need to employ an additional communication means besides the existing Web Services technology, and no need extra polling packets. Alarm/event and COV are sending as Web Service requests, every request will get a corresponding response as confirmation. Thus every notification is acknowledged with confirmation. This provides the reliability of communication. Actually, the chain-used IBmanager

installations discussed above are of this kind of application.

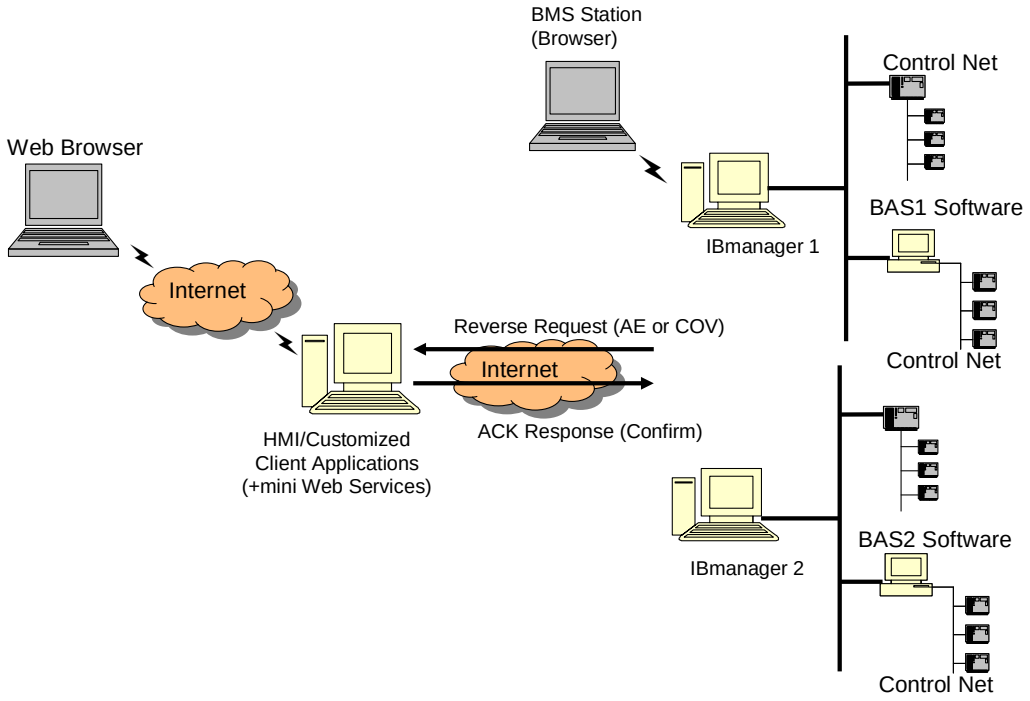


Figure 3.9 Transport notifications by “Peer to Peer” method

Method 2: “Piggybacking” Technology

Another method can be called “piggybacking” technology, it is to “piggyback” the notification information when the IBmanager responds its client with the requested information as Figure 3.10. Normally, when the client requests information, the IBmanager will send back the corresponding response information. In this “piggybacking” design, besides the desired response of the request the IBmanager will piggyback additional notification information to client meanwhile. The notification information is piggybacked within the response packets. This piggybacked information will be received and processed by the client. In order to ensure the reliability, the client - piggybacked information receiver will send

confirmation request to the IBmanager. Thus, the reverse message transferring is realized.

However, the frequency of the issued requests by client are not time-deterministic, how does the IBmanager send notification to the client when no request is made from the client to the IBmanager? In order to keep the notification messages can be transported timely, the client is designed to initiate null-contented requests frequently even when it has no real information to be retrieved from the IBmanager. So a timer is defined, if the client has no real request to the IBmanager, a null-contented request message will be sent to poll the IBmanager for possible notification messages when this timer event is triggered.

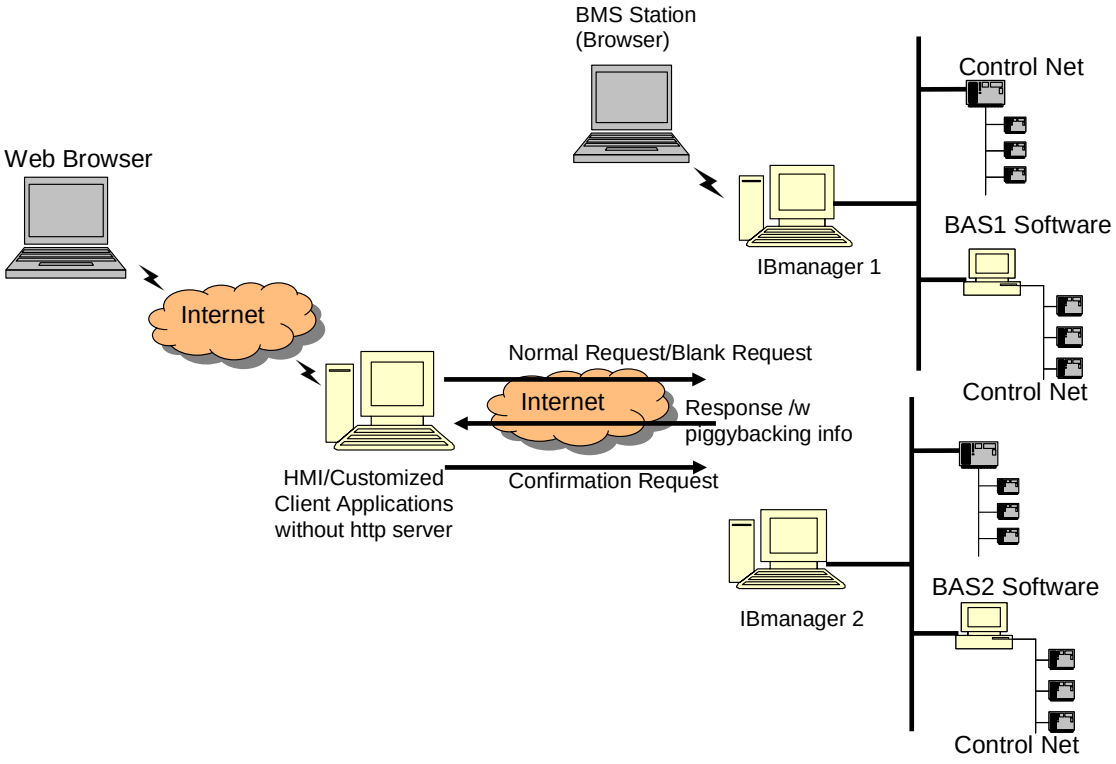


Figure 3.10 Transport notifications by Web Services “piggybacking” technology

Comparison between these two methods

Regardless of which method, it should have a subscription process in advance to let the IBmanager know which information the client is interested in. Only when this subscription process is settled, the IBmanager can send the correct information to the right receiver.

Both these methods achieve the bi-directional information transportation. The advantage of “Peer to Peer” method is that the A/E and COV information will be conveyed without latency and no additional packet added, its disadvantage is that it need deploy a Web server in the client computer, which will add system load and is not always feasible, for example, when firewall blocks the port. The advantage of “piggybacking” technology is that it need not change the architecture of the system, need not deploy a Web server at the client computer. Its disadvantage is the latency the timer-triggered-request leads to. One can adjust the latency by changing the interval of the polling timer. Compared to the “Peer to Peer” method, the acknowledge process need additional request packet to ensure the communication, so it increases the network traffic.

Table-2 compares the characteristics of these two methods. The choice of these two methods depends on the application environment.

Table-2 Comparison of the characteristics of these two methods

	additional Web Server / Service on client side	additional traffic	latency
“Peer to Peer” method	Need	No	No
“piggybacking” method	No	yes(acknowledgement)	yes(triggered polling)

3.3.4 Reliability of the Reverse Communications

By the two methods of reverse message transportation discussed above, the alarm/event and Change-Of-Value (COV) is transported from the IBmanager to client. However, the reliability of the communication must be considered. For example, if COV message related to one data point is missed and no more COV about the data point occur in a subsequent duration, the information about the data point in the client will not be updated to correct value in this duration. That is, the client will keep a wrong value until receiving the next COV massage. In order to avoid this, these two reverse communication methods above are designed with confirmation. After the sender receives the return confirmation from the receiver, it thinks the transportation and the process is finished, otherwise it thinks the transportation or the process fails and sends the packet again.

3.3.5 Synchronous/Asynchronous Communication

HTTP is a synchronous protocol since a client connects to a server and it submits some information and waits for a response. The communication based on HTTP is

synchronous communication, since no callback method can be used to notify the arrival of the reply. Web Service is based on HTTP protocol; it is a synchronous protocol certainly.

However, based on the reverse message transportation methods presented above, an asynchronous communication method of Web Service can be achieved as Figure 3.11. When the client sends a request, the IBmanager responds with a delay-indication message indicated that the response is delayed and will be sent later. After receiving this delay-indication response, the http server process in the client can take other tasks and need not wait for the “true” response. When the IBmanager finishes executing the task and the client-requested information is ready, the IBmanager will “callback” the client and send back the “real” response. The “callback” can be any one of the two methods discussed above. Thus an asynchronous communication method of Web Service is achieved.

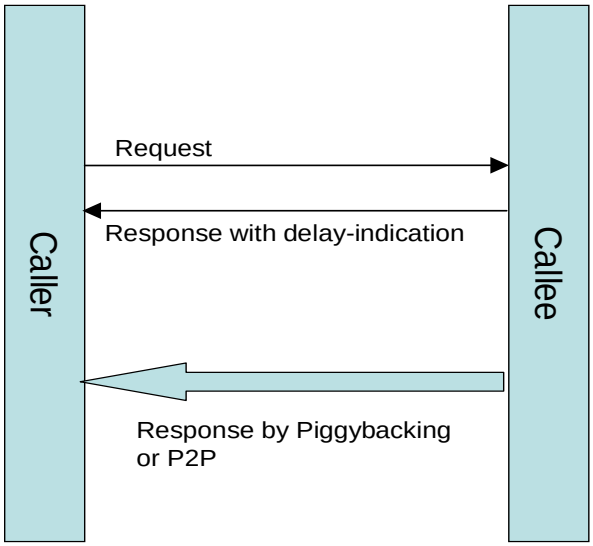


Figure 3.11 Asynchronous Web Service communication

3.3.6 Point Aggregation/Grouping

The communication will be much more frequent if the operation related to every data point is sent in a standalone data packet. This will lead to great overhead for the network traffic. Some methods are introduced to decrease the traffic, for example, data points grouping and aggregation.

Grouping

This method is used in the *"piggybacking"* technology. In the design, all the subscribed data points with the same update interval will be grouped into one group. The subscribed data points with the different update interval will be grouped into different group when subscription. Once receiving normal request or null-contented request, the groups' management software (the IBmanager) will decide update information of which group will be sent to the client according to its update interval as Figure 3.12. The group with smaller interval will send notification more frequently. When the condition is satisfied, the update information of all the data points in the group to be sent will be transferred in packets (information may be fragmented by http handler).

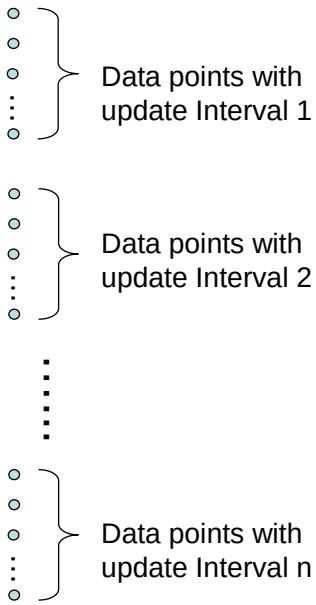


Figure 3.12 Grouping update information of data points by interval

Aggregation

Another method is to aggregate the information of data points, it is suitable for "*Peer to Peer*" method discussed before. In this method, the time to send information is decided by the IBmanager, in order to decrease the frequency of sending data, the IBmanager will not sent out the update information immediately it occurs. All the update information occurred within the *WaitingTime* (a variable which user can adjust) duration will be aggregated into one package to send as Figure 3.13. This aggregating method doesn't maintain a steady points list, just aggregates the occurred update information which to be sent in the *WaitingTime* duration together. In this method, the most unfortunate data point will have a latency value of *WaitingTime* in maximum. If *WaitingTime* is set to zero, the packets will be sent out immediately.

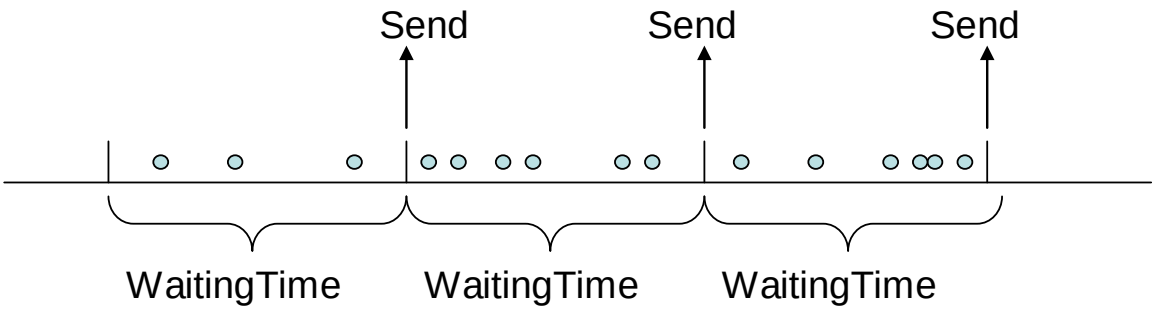


Figure 3.13 Aggregating update information within a duration

3.4 Heterogeneous Databases Integration by Web Service

The integration platform needs retrieve configuration and historical data from local or other remote databases. The access to local database is easy with the flexible and standard interfaces. The integration of remote heterogeneous databases is a challenging task. This section will focus on the integration of remote heterogeneous databases.

3.4.1 Remote Heterogeneous Databases

These remote databases from which the integration platform needs retrieve data from are constructed and maintained by various providers with different technologies. Some databases are open for users to direct access with popularly-used RDMS (for example, Microsoft Access, Microsoft SQL Server), some databases can be accessed with SQL-based operation query language, some databases can be accessed by API invocations, and some use proprietary export tools to export to open format data (for example, Excel format or Access database).

3.4.2 Integration of Remote Database by Web Service

In order to integrate various remote databases by Web Service, a Web Service wrapper was developed for remote database and enables the remote database management software as a Web Service provider. In this design, the Web Service wrapper provides Web Service interface which a remote client can call as Figure 3.14. The Web Services interface that has been described in the WSDL file typically bind to an underlying program or middleware application that handles the incoming request, executes a database query, and returns the output of the query. It is then up to the Web Service wrapper to encode the output of the application in SOAP message, and send the message back to the requesting client.

At the forefront of the Web Service wrapper is the SOAP engine for handling incoming SOAP request messages. Web Service wrapper demarshalls and decodes SOAP messages, and forwards the decoded tasks to the “DB Query Module” that executes corresponding tasks on the database using an ODBC driver. After executing an SQL statement through ODBC, the data return is passed back to Web Service wrapper from the “DB Query Module”, to where the data is marshalled into a SOAP response message and return to the Web Service requester [73].

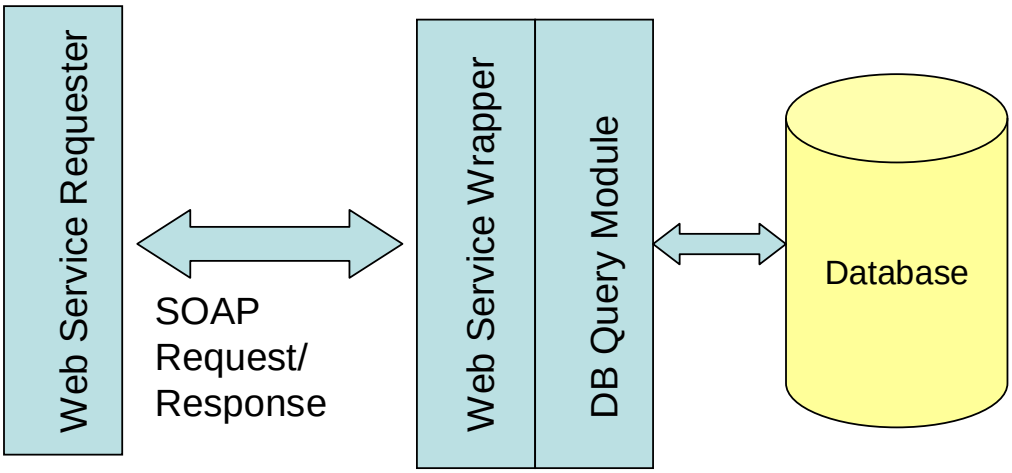


Figure 3.14 Architecture for accessing database by Web Service wrapper

3.4.3 Data Compress for Web Service Transportation

On some Web Services request to database, the SOAP response could be a very large dataset. For instance, after it is serialized to XML, a dataset that contains all columns of the table "orders" from sample Northwind database in the SQL Server installation is about 454 KB. If one creates an application that retrieves this dataset by invoking a Web Service, the SOAP response would contain all of that data. The transfer of data through a network, and especially the Internet, the bandwidth associated with data transfer remains a bottleneck in many distributed systems. One solution to this problem is to acquire more bandwidth, but this is not always practical. Another solution is to minimize the amount of data to be transferred by compressing it. When the content is text, its space can be reduced up to 80% after compressing. This means that the bandwidth requirements between consumers and servers decrease at an analogous percentage [74]. Since dataset is serialized to XML and XML content is

text, the compressing can greatly reduce the data to be transferred.

There are different ways to compress SOAP messages. SOAP Extensions is an option for this aim. SOAP extensions is a Microsoft ASP.NET WebMethod interception mechanism that can be used to manipulate SOAP requests/responses before they are sent on the wire. Using SOAP extensions, the size of SOAP messages traveling on the wire can be reduced greatly when a consumer invokes a method from a Web Service. Developers can write code that executes before and after the serialization and deserialization of messages.

In most cases SOAP requests are much smaller in size than SOAP responses (for example, a large dataset), the increase in performance resulting from compressing SOAP requests would be insignificant, so only compressing SOAP responses is considered in the example below. In order to compress our Web Service's SOAP responses, two things need to be done:

- Compress the SOAP response message after serialization on the server
- Decompress the SOAP response message before deserialization on the client

As Figure 3.15 on server side and AfterSerialize stage the SOAP response is compressed and travels on the wire as a compressed SOAP message, on client side and BeforeDeserialize stage the compressed SOAP message is decompressed in order deserialization process to follow successfully [74].

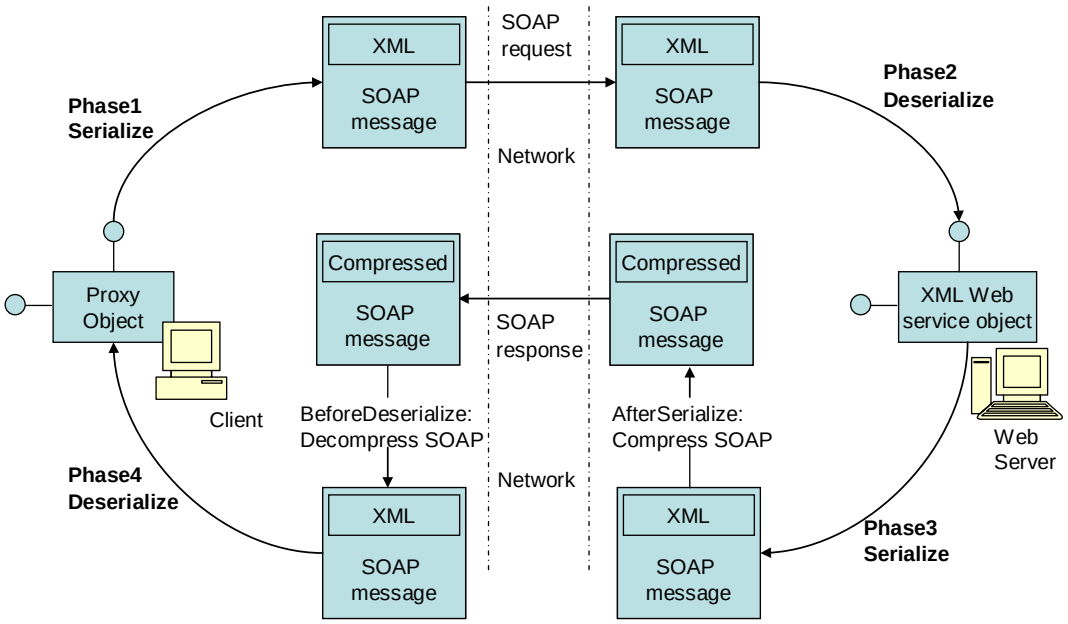


Figure 3.15 SOAP message compressed on AfterSerialize stage (server side) and decompressed on BeforeDeserialize stage (client side)

CHAPTER 4 OBJECT MODEL OF THE IBMANAGER

4.1 Overview of System Architecture

4.1.1 Software Architecture

Figure 4.1 illustrates the component architecture of IBmanager platform. The middleware platform can be divided into several parts, i.e., the Drivers Pool, the Database Agent, Application Server, Web Services interfaces and Web Server, Client or BMS HMI.

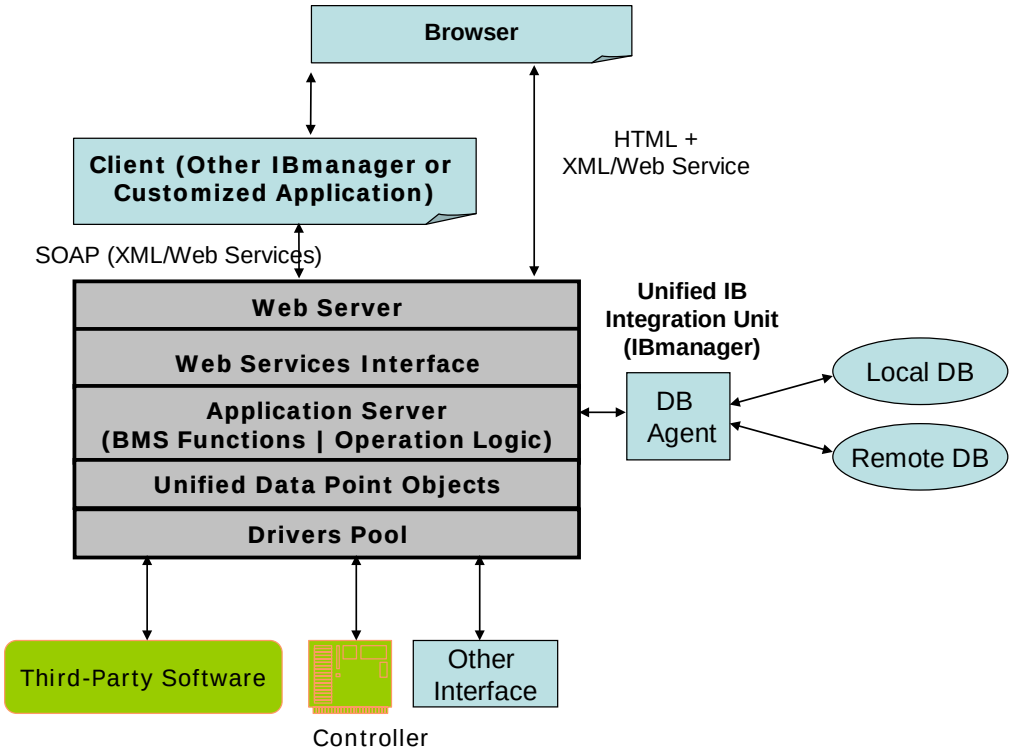


Figure 4.1 Component architecture of the IBmanager

Driver Objects are responsible to manage various drivers, realizing real time data access, and interfaces to third-parties applications. In the Drivers Pool, BASs

from different manufacturers and with various protocols are connected. The connections may be conducted by various communication ways, including:

- Communication interfaces or APIs provided by BAS software or
- Protocol driver accompanied with devices or
- Customized communication interfaces converted from other drivers or other interfaces.

Usually the third-parties' BAS software provides, at least, one high-level interface, including OPC server, DDE, COM/DCOM or other interface. For example, Honeywell EBI providing OPC Server interface [80], LNS providing DDE interface [83]. All these high-level interfaces can be connected into the IBmanager.

A unified **Data Point Object** model is presented. All the real time data read from Driver Object are encapsulated as the unified **Data Point Object** model, Data Point Objects are the core part of IBmanager. It supervises the real time data value from sub-systems/devices, transfer events/messages to BMS function objects to achieve the high-level BMS functions, including Schedule Object, Interoperation Object, Alarm/Event Object, Session Object, and Database Agent Object. For example, values of the data point are recorded into a historical database if the data points are configured as history-recording data points.

Application Server (App. Server) realizes various BMS functions and operation logic. BMS function components executes the tasks of real time data access, alarms & events process/dispatch, historical data access, scheduling, trending and scripting logic. All these functions are encapsulated as different objects.

Database Agent provides a unified interface to access local database and remote

database. It shields the differences between local database and remote database to provide a unified access interface.

The functions provided by the IBmanager for public invocation are wrapped as public **Web Services interfaces** which can be invoked by other applications (HMI or other client applications) or other chain-type used IBmanager installations.

The **Web Server** deployed in the IBmanager is not only used for web page storage and access, but also used for http protocol parser since transportation method for Web Services adopted here is http protocol. It is responsible to parse the Web Service request from http messages and forward the request to the Application Server. These Web Services interfaces are public to provide services. The Web pages in the Web server provides Web human machine interface (HMI) to users, the components in the Web pages communicate with the IBmanager by Web Services technology.

4.1.2 Object Model

In typical BAS/BMS, several important functions should be implemented, such as real time data access, alarm/event, historical data, and trending, scheduling, network management [7]. In this design, the function blocks of the IBmanager are encapsulated as objects as Figure 4.2. The communication with sub-systems/field devices is accomplished by **Driver Objects,** information management of data point is accomplished by **Data Point Objects,** historical data access is managed by **Database Agent Objects**, alarms & events is dispatched and managed by **Alarm/Event Objects**, interoperation is charged by **Interoperation Object**, scheduled tasks are arranged by Schedule Objects, the communication with clients is responsible by

Session Object. Figure 4.2 shows functions blocks and objects in this middleware platform.

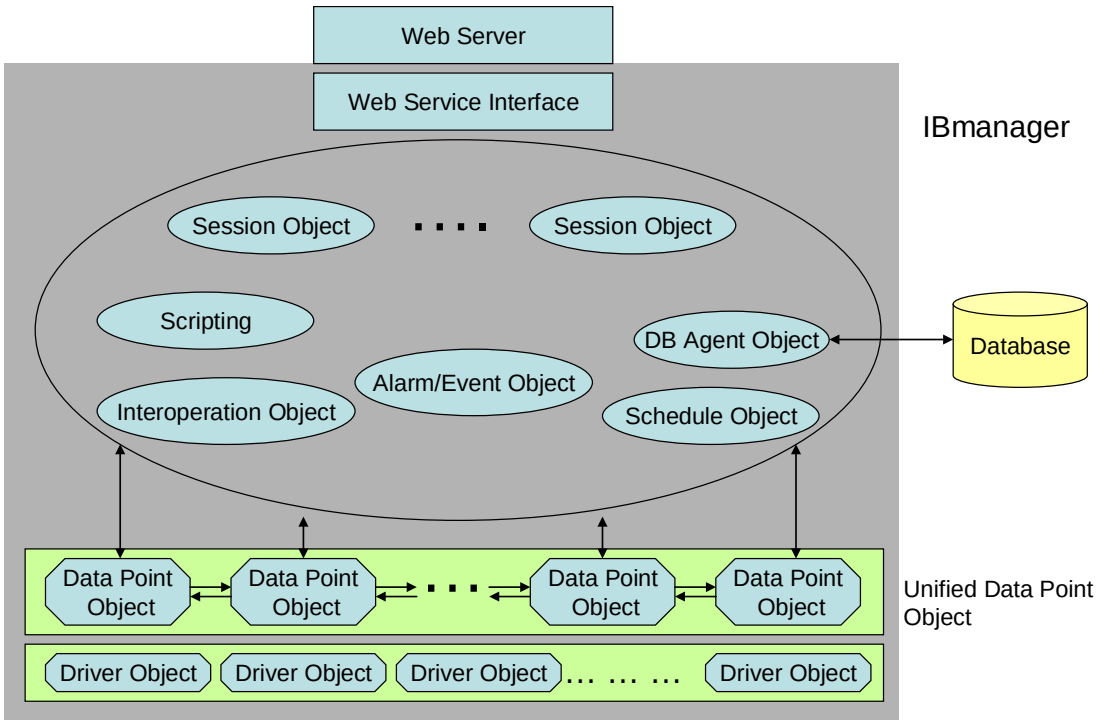


Figure 4.2 Object model of the IBmanager

4.2 Driver Object Model

4.2.1 Driver Object Model

The driver is encapsulated as object as Figure 4.3. The Driver Objects can be generated dynamically as need. Adding driver for new sub-system is easy by creating and loading a Driver Object as the driver specification (Appendix A) defined. Meanwhile, user can unload a Driver Object easily when the sub-system corresponding to this driver is not connected any longer. This will avoid unnecessary overload of the IBmanager. By the Driver Object, various data points from diverse

products or protocols are converted to the unified Data Point Object model.

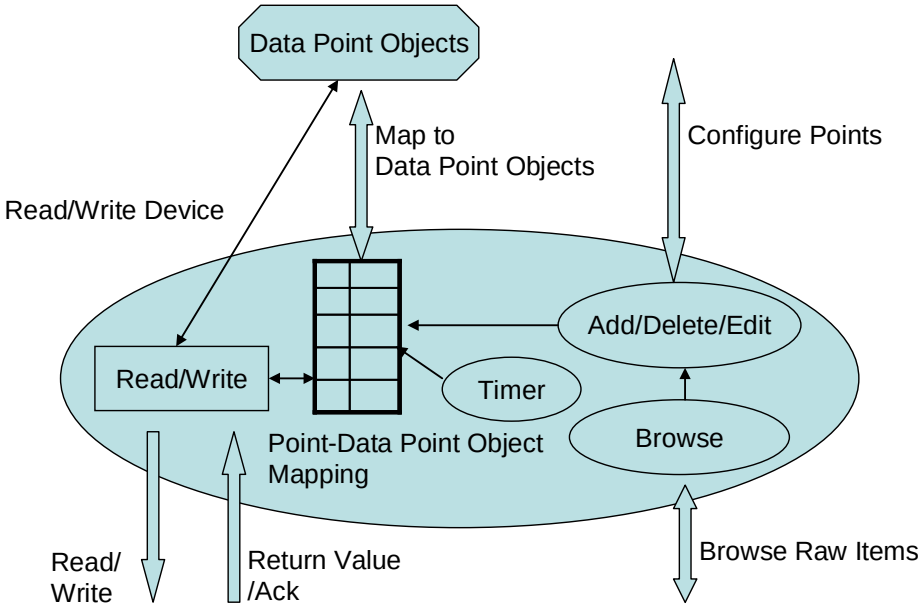


Figure 4.3 Driver Object model

4.2.2 Elements of Driver Object

Configure Points

Besides the method to read/write the value of data points, the Driver Object should support the methods to **Browse/Add/Edit/Delete** data points as well. The browse method is to browse the raw data points of the connected systems/devices. The original BAS or BMS may have a great collection of data points, usually the new integration software is only interested in part of the entire data point collection. From the data points browsed, users can select data points interested and add them to IBmanager system with “Add data point” method, at the same time ignore the data points which users don’t care about. This keeps this system is flexible and scalable. In

order to meet the users' habit, users can assign an human-friendly alias for the added points, after that, the high level operations will be conducted to this alias instead of raw item name. Users can edit/delete the added data points by "Edit/Delete data points" methods as well.

Mapping of Points - Data Point Objects and Value Update

In traditional way, when the Driver Object receives the request related to its data point, it will search the database (if the values of data points are stored in database) to get the value of the data point or just begin to read the value from field devices. However, the response performance from the database query or field devices may be not good. It will be a serious problem when the driver maintains a great volume of points.

In this design, the Driver Object will maintain a **memory** collection table, which maps the selected data points' name to Data Point Objects. The current values of data points are saved in the corresponding Data Point Objects. In order to keep the values update, timers in the Driver Object are used to trigger the event to update the value of data points frequently, the updated values will be saved to the Data Point Objects in advance. Usually the upper-level functions or objects will read the values of data points saved in the Data Point Objects, instead read directly from the subsystems/devices by drivers. This will leverage the speed of the operations. In this model, the timer as a polling timer is used to trigger the "read data points" operation to update the value of data points frequently.

Besides this timer-triggered reading in advance, “ReadDevice” method of Data Point Object is defined as well for direct access to the data source system as an option. Data Point Object can issue out a request to read value from the field device/sub-systems directly if necessary.

4.3 Common Data Point Object Model

A common Data Point Object model is designed as the intermediate between driver layer (Driver Objects) and upper layer. The different data models from all sub-systems are converted into the common Data Point Object model. These common Data Point Objects are the core part of IBmanager; all the high level functions are based on these Data Point Objects. The high level functions include scheduling, interoperation, alarm/event, database access, session management. The communication between Data Point Objects and high level function objects is mainly conducted by events. Usually, trigger term is set in the Data Point Object by high level function objects, the Data Point Object fires event when the trigger term meets. The event will be captured and handled by the corresponding high level function objects.

4.3.1 Data Point Object Model

The Data Point Object maintains the configuration information of the data point, reads/writes values triggered by the timer in the Driver Object, raises events, and dispatches the event messages to high-level BMS functions objects.

The Data Point Objects are instantiated and maintained by the corresponding Driver Object. When the Driver Object is constructed, all these Data Point Objects belonged to the Driver Object are instantiated by the Driver Object according to its configuration table. The Driver Object will invoke “ReadValue” method of Data Point Objects frequently with triggered by a polling timer, thus the values of these data points are kept updated. The main communication methods to the high-level function objects are sending events/messages to these high-level objects when they subscribe the corresponding events/messages.

Figure 4.4 shows the application model of the Data Point Object. There exists communications between the Data Point Objects besides the communication between Data Point Objects and upper layer entities. These communications are used to realize the interoperation/interlock among the Data Point Objects. One Data Point Object can operate another Data Point Object when necessary.

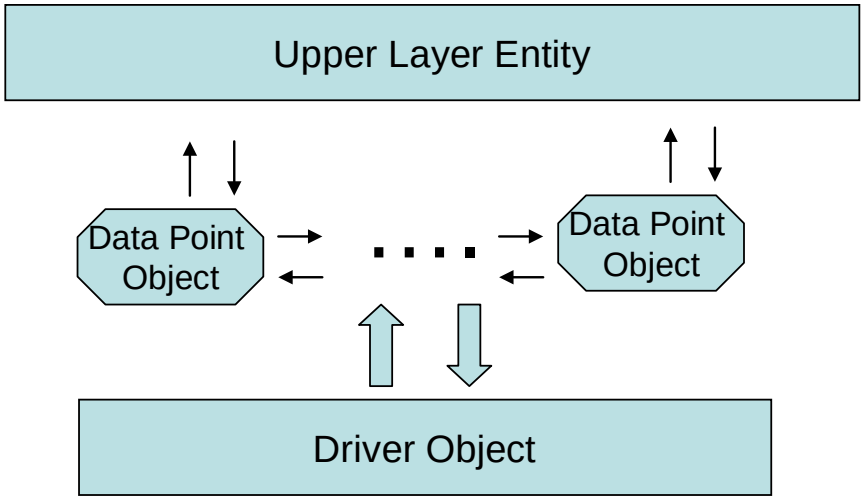


Figure 4.4 Convert data of various protocols to common Data Point Objects

4.3.2 Elements of Data Point Object

Every Data Point Object contains properties, methods, events defined as Figure 4.5.

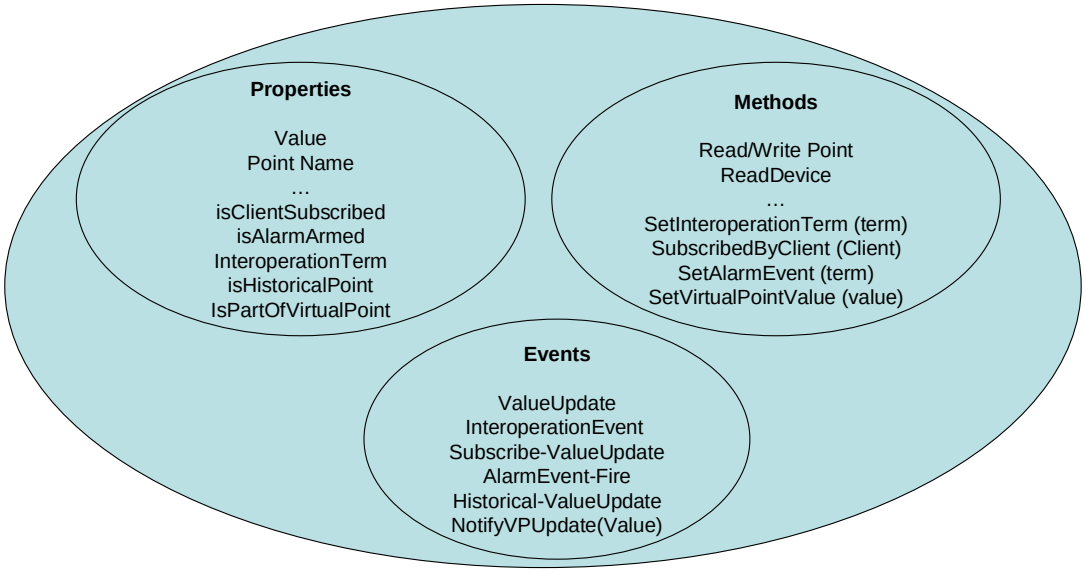


Figure 4.5 Common Data Point Object model

Every Data Point Object has properties and events as below:

Properties:

Value

Unit

Point name

Pre scale

Deadband //deadband for value-update event firing

Alias Name //a human-friendly name

Permission //only readable or readable/writeable

InteroperationTerm //Interoperation event will be fired when the term meets

IsClientSubscribed //the Subscribe-ValueUpdate event is fired when it is true and the value-change is beyond deadband

IsAlarmArmed //the alarm/event will be fired when the AlarmTerm is met and this property is set true

AlarmTerm //the term for firing AlarmEvent-Fire event

IsHistoricalPoint // whether the data point is of history-recording

HistoricalInterval //the interval to record this data point

IsPartOfVirtualPoint //a NotifyVPUpdate event will be fired or synchronous method to be executed to calculate the virtual point when this property is true and the data point has an value update

Methods:

Read Point

Write Point

ReadDevice //read data point directly from sub-systems/devices

SetInteroperationTerm (InteroperationTerm) //the InteroperationTerm may be composite term

SubscribedByClient (Client)

SetAlarmEvent (AlarmTerm) //the AlarmTerm may be composite term

SetVirtualPointValue(Value) //synchronous method to update virtual data point

Events:

ValueUpdate

InteroperationEvent

AlarmEvent-Fire

Subscribe-ValueUpdate

Historical-ValueUpdate //event to notify to record historical data

NotifyVPUpdate(Value) //event to notify virtual data point to re-calculate

In order to easily introduce these elements, the main elements can be categorized to groups according to the functions.

i. General/Basic:

The object properties *Value, Unit, Point name, Pre scale, Deadband, Alias Name, Permission* and event *ValueUpdate* are for generic use.

Deadband and *ValueUpdate* are used to trigger value-update notifications. The event *ValueUpdate* is the basic event of some high-level events which includes "*InteroperationEvent*", "*Subscribe-ValueUpdate*", "*NotifyVPUpdate(Value)*" and "*Historical-ValueUpdate*".

ii. Interoperation:

The "*SetInteroperationTerm (InteroperationTerm)*" and "*InteroperationEvent*" are used to trigger to execute the interoperation action. On initialization, the Interoperation Object will invoke the "*SetInteroperationTerm (InteroperationTerm)*" of interoperation-related data points to set the term to trigger interoperation. After that, when the term is satisfied, the "*InteroperationEvent*" event will be fired. The Interoperation Object will catch the event and make the corresponding operations.

iii. Client Connection Session:

The "*SubscribedByClient (Client)*" and "*Subscribe-ValueUpdate*" is used to notify the ValueUpdate to the Session Object. Once a client connects to the IBmanager, a Session Object will be instantiated. This Session Object will invoke the "*SubscribedByClient (Client)*" of related data points to subscribe the *Value-Update* event. After that, when the general *Value-Update* is fired, the Data Point Object will judge whether some Session Objects have subscribed this event, if yes the

"*Subscribe-ValueUpdate*" event will be fired and processed by the Session Object. Thus the Session Object will know the *Value-Update* of the interested data points only. Its advantage is that every session will only monitor the data points it is interested in. This will greatly decrease the system load and net traffic.

iv. Alarm/Event:

The "*SetAlarmEvent (AlarmTerm)*" and "*AlarmEvent-Fire*" are used to set the alarm/event term and fire alarm/event when the alarm/event term is satisfied. On initialization, the Alarm/Event Object will invoke the "*SetAlarmEvent (AlarmTerm)*" of alarm/event-related data points to set the term to trigger alarm/event. After that, when the term is satisfied, the "*AlarmEvent-Fire*" event will be fired. The Alarm/Event Object will capture the event and make the corresponding operations, for example, judging whether a combinational alarm/event term is met, distribute the alarms/events to related clients. Every alarm/event has corresponding receivers, so every client can subscribe only the alarms/events it cares about.

v. Historical Recording:

"*IsHistoricalPoint*", "*HistoricalInterval*" and "*Historical-ValueUpdate*" are used to notify the Database Agent Object to record the value of the data point. On initialization of Database Agent Object, the Database Agent Object will set the "*IsHistoricalPoint*" property to "True" and "*HistoricalInterval*" for the historical data points. After that, when there is general *ValueUpdate*, the historical Data Point Object

will judge the time and interval to decide whether the "*Historical-ValueUpdate*" event will be fired. If the "*Historical-ValueUpdate*" event is fired, the event will notify the Database Agent Object to record the value.

vi. Virtual Data Point:

Virtual Data Point is a kind of Data Point Object whose value is based on the calculation of the values of other data points existed. The Virtual Data Points have the same elements with the common Data Point Objects but without corresponding physical control or monitor data points. It is useful for some intermediate calculation results, for example, enthalpy is the multiply of humidity and temperature. So its value is related to other data points and should be re-calculated when the related data points have *value-update* event. "*IsPartOfVirtualPoint (VirtualPoint)*", "*SetVirtualPointValue(Value)*" together with "*NotifyVPUpdate(Value)*" event is used to automatically update the value of virtual data point. It will be discussed in details later.

4.3.3 N+m Architecture

By transforming the different data points from diverse systems into normalized Data Point Objects, the IBmanager provides a common data format for upper layer applications. The data format what the upper layer applications deal with is only the common data format. When upper layer applications with number N are deployed, it forms an N+m architecture that provides substantial benefits over traditional

drivers-based integration methods that suffer the complexity of N*m architecture as Figure 4.6. Any sub-system/field device normalized by the IBmanager immediately becomes compatible with any other sub-system connected to the platform. This benefit provides true inter-system interoperability and uniform data presentation to enterprise applications. As a result, N+m architecture greatly decreases development and maintenance labor of developers and engineers.

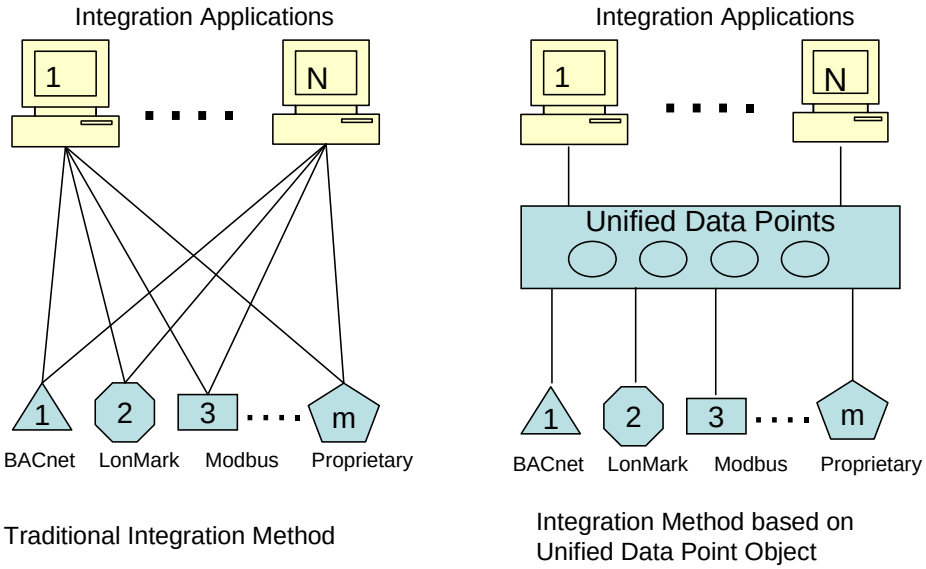


Figure 4.6 Architectures of traditional and designed integration methods

4.3.4 Event-driven Model among Components

Usually the high-level function components have two different methods to know the update of the values of data points. In one method, the high level function components poll the drivers about the data point value frequently. In this case, the IBmanager maintains a great collection of data points and polls the values of data points frequently as the configured intervals, regardless of whether the values are

changed. This method will greatly increase the burden of computer.

The second method uses an event-driven model instead of polling model. Every Data Point Object in the Driver Object has a “deadband” property. Only when the value of the Data Point Object has changed beyond the defined “deadband”, the Data Point Object will fire an event named “*ValueUpdate (point name)*” event. The value changed within the scope of deadband will not fire the event. These “*ValueUpdate (point name)*” events then trigger various specific events when the trigger terms configured in the Data Point Objects are met, such as “*InteroperationEvent*”, ”*Subscribe-ValueUpdate*”, “*AlarmEvent-Fire*”, “*Historical-ValueUpdate*”. These specific events trigger the specific operations in high-level function objects. Main communications among the components in the IBmanager are conducted by events/messages.

The trigger terms of the Data Point Objects are configured by high level function objects. This is similar to “subscribe”, when one high level function object cares about specific information of a data point, it will “subscribe” or “declare” what it is interested in. when the term is met, the Data Point Object will notify the high level function objects what happens.

4.3.5 XML Hierarchy and Object Encode

XML is easy to be used to encode the hierarchy data. The object can be serialized as XML message to transfer. The elements of object can be coded as xml sub-element,

if this element is also an object, then it has also some sub-element. Below is an example of serializing object to XML message.

```
Class Data
{
   int value;
   String name;
   Date updated;
}
```

This object can be serialized as:

```
<Data>
   <Value style='int'>123</Value>
   <Name>Temperature</Name>
   <Updated>2006-10-04T17:23:23.123+0800</Updated>
</Data>
```

That’s very natural encoding method. On the counterpart side, the XML message will be decoded to the original object. By this method, complex data structure as object can be transferred as XML messages.

4.3.6 Virtual Point Object/Data Fusion

The Virtual Data Point Object can be used to realize complicated calculation and some kind of data fusion. More importantly, the Virtual Data Point Object is identical

to common Data Point Object, it can be manipulated alike common Data Point Objects by other part of the IBmanager, like display and alarm/event.

On initialization of Virtual Data Point Object, the Virtual Data Point Object will set the "*IsPartOfVirtualPoint*" property of related data points composing the Virtual Data Point Object to "True" and pass in the name of the Virtual Data Point Object. After that, the value-update process of the Virtual Data Point can be conducted in two different ways. One is synchronous operation method as Figure 4.7, when the related data points fire general *ValueUpdate*, they will invoke "*SetVirtualPointValue (Value)*" method to transfer their changed values to the Virtual Data Point if "*IsPartOfVirtualPoint*" is true. The Virtual Data Point will recalculate its value according to the updated values of the related data points. Thus, the value of the Virtual Data Point will be updated when any related data points have fired *ValueUpdate* event. This method is just like a "callback" mechanism. The advantages are more direct and without events being processed by a lot of Data Point Objects.

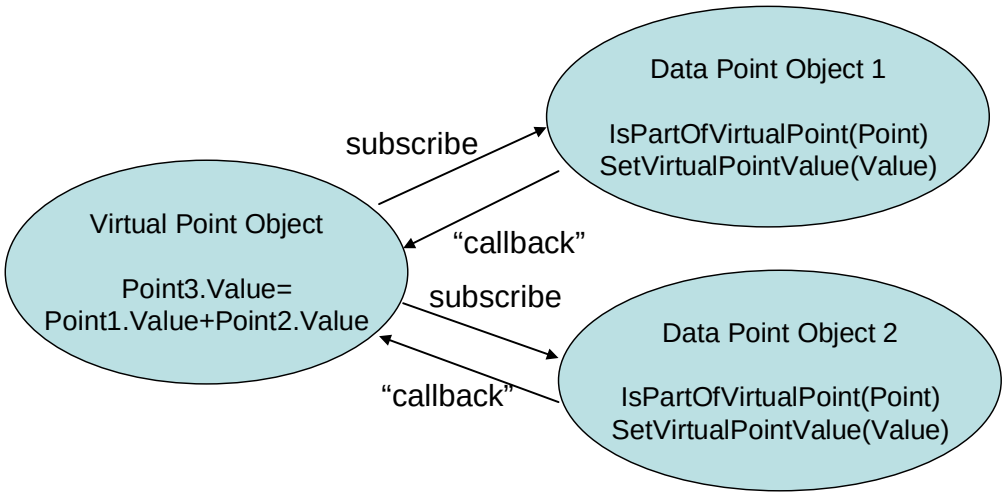


Figure 4.7 Value update of Virtual Data Point Object

The second method is event notification method based on "*IsPartOfVirtualPoint(VirtualPoint)*" together with "*NotifyVPUpdate(Value)*" event. When the related data points fire general *ValueUpdate*, "*NotifyVPUpdate(Value)*" event will be fired, the "*NotifyVPUpdate(Value)*" event will be caught by the Virtual Data Point, and the Virtual Data Point will trigger the re-calculation of its value.

4.4 Schedule Object Model

The unified Schedule Object maintains all the scheduling tasks with Schedule Table as Figure 4.8. The Schedule Table maintains a collection to contain all the *trigger time – task* pairs and task repeat information. These pairs and task repeat information are defined by users and stored in database. When the IBmanager starts, they are read into Schedule Table of this Schedule Object as Figure 4.9. The time gap to the earliest trigger time in this table is set as interval of the timer. Thus, the interval of the timer is dynamic, the timer will sleep until the earliest trigger time without not necessary event- firing. When the timeout event of timer is fired, the corresponding task is activated. After one *trigger time – task* pair is fired with event, the new trigger time of the task will be recalculated as the repeat parameter if it has. The time gap of the new earliest trigger time is set to the timer as interval. Thus, only one timer is kept for all the scheduled *trigger time – task* pairs. Since all the schedules uses the same sole timer, and it uses dynamic interval method instead of fixed interval, this method can decrease overhead of the system greatly.

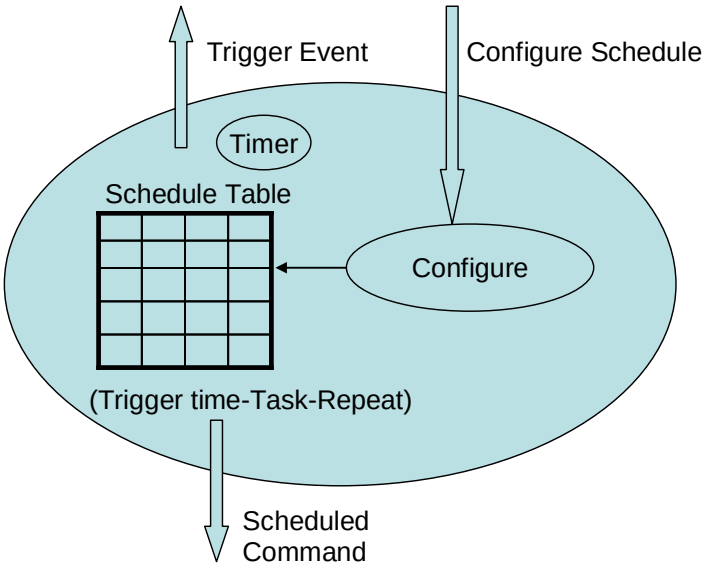


Figure 4.8 Schedule Object model

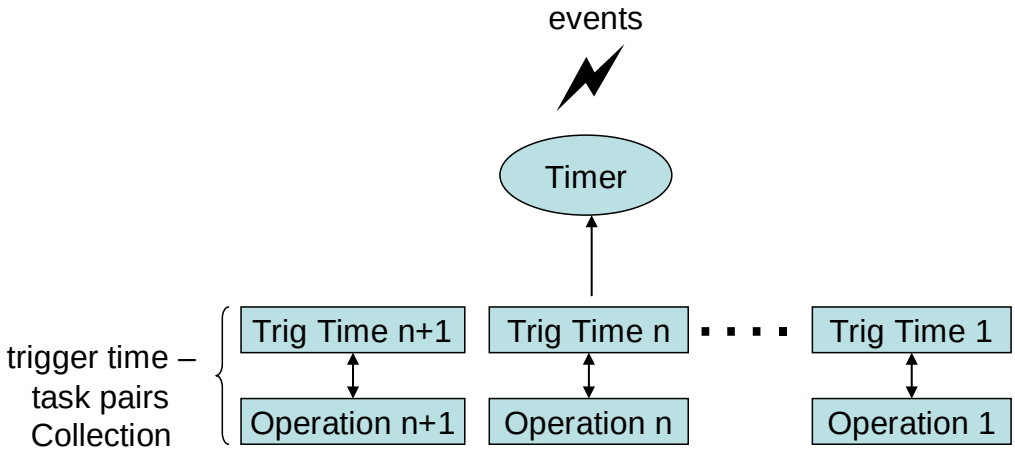


Figure 4.9 Timing model of Schedule Object

The time restrict of the timer is 65535ms (about 65s) in Windows, however, the interval of the schedule may be a long time, so a multiply number is defined for long interval. This number will decrease one when the timer fires short interval event. When the number meets zero, the timer will fire long interval event. So the long interval is multiply number times of the short interval.

The overall task is finished in the Schedule Object. In order to provide a chance for some possible higher layer operations, such as user-defined scripts, these events will be thrown outside the Schedule Object and can be captured by higher layer functions**.**

4.5 Alarm/Event Object

In the Alarm/Event Object as Figure 4.10, the Alarm/Event Term Table keeps the trigger terms of alarms/events, which are loaded from database at startup. They can be configured and modified by users' configure commands. After modifications, the updated information will be saved into the Alarm/Event Term Table and database.

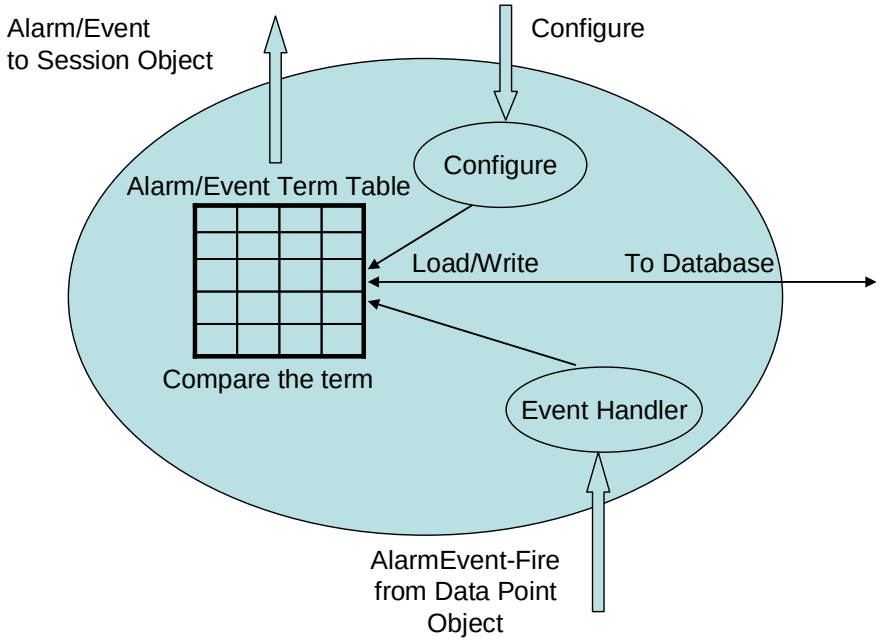


Figure 4.10 Alarm/Event Object model

Alarm/Event Object sets the "*isAlarmArmed*" property and invokes "*SetAlarmEvent (term)*" of Data Point Object to configure its trigger term according

to the Alarm/Event Term Table. The Alarm/Event Object monitors AlarmEvent-Fire event from Data Point Object, when the event is fired and captured, it will save the information of the event into the Alarm/Event Term Table. Since the AlarmEvent-Fire from Data Point Object is only related to one Data Point Object, however the Alarm/Event term may be composite and relate to more than one data point, so it need check other conditions to decide whether the composite Alarm/Event term is met. If the Alarm/Event term is met, an Alarm/Event will send to corresponding Session Object. At the same time, this Alarm/Event will be logged into database.

4.6 Interoperation Object

4.6.1 Two Different Interoperation Environments

The interoperations may occur at two kinds of environments. One occurs between the BASs managed in one IBmanager installation as Figure 4.11, another is across different IBmanager installations as Figure 4.12. These two different application cases are described as below.

The first case is the interoperation among sub-systems which are connected to the same IBmanager installation. This interoperation is realized internally by the IBmanager. The second case is the interoperation among IBmanager installations, it may be across the Internet, involving Web Service communication. However, based on the unified data model and the unified communication interface, the interoperation methods in these two cases are realized identically.

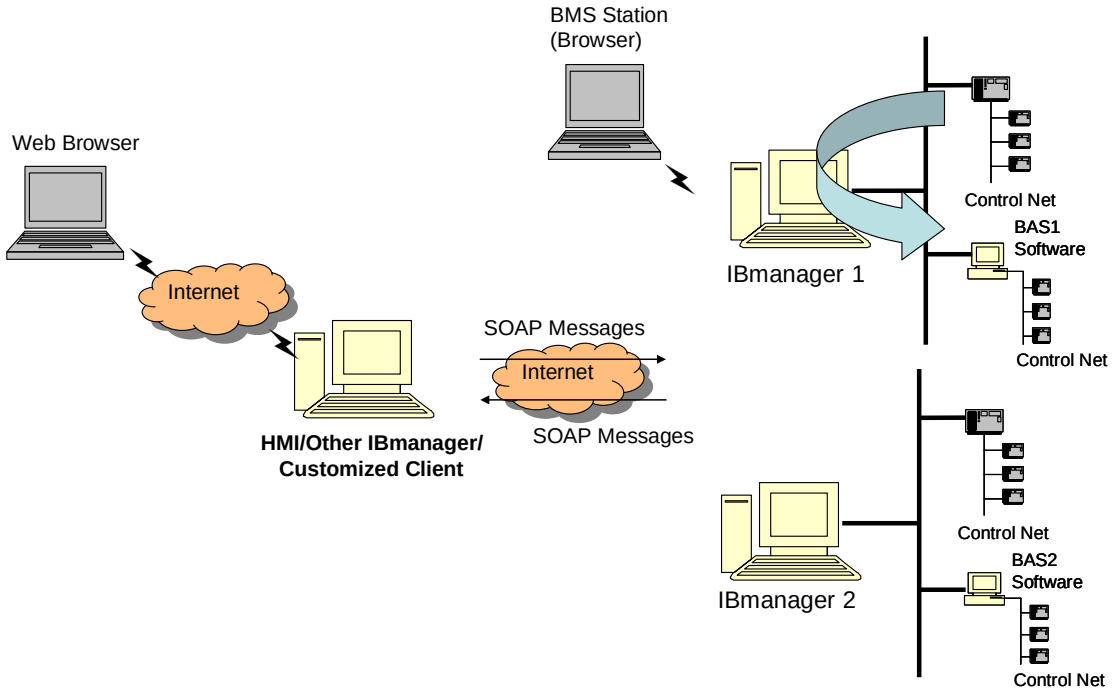


Figure 4.11 Interoperation between sub-systems within the IBmanager

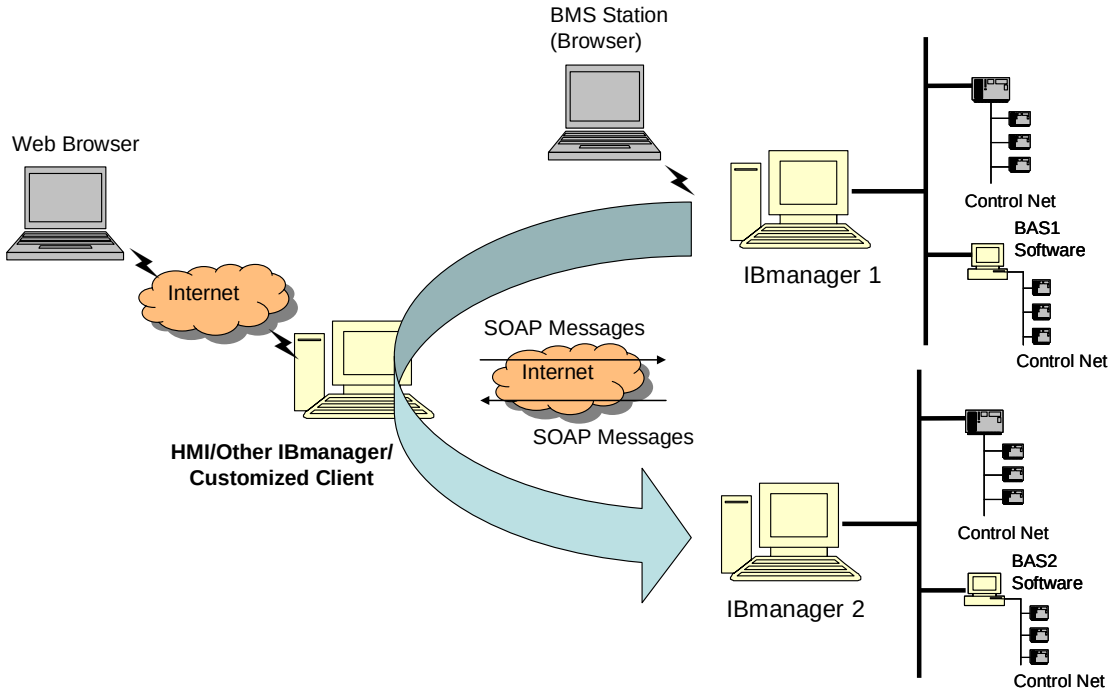


Figure 4.12 Interoperation across IBmanager installations

4.6.2 Interoperation Object Model

How does IBmanager know which term will result in interoperation operation? Traditional method is comparing the every interoperation trigger term with the real situation as Figure 4.13, even the data points are not related to interoperation trigger term. It will consume much CPU power and lead to bad performance if this method is used in great project with a great amount of points. So a new Interoperation Object model is defined.

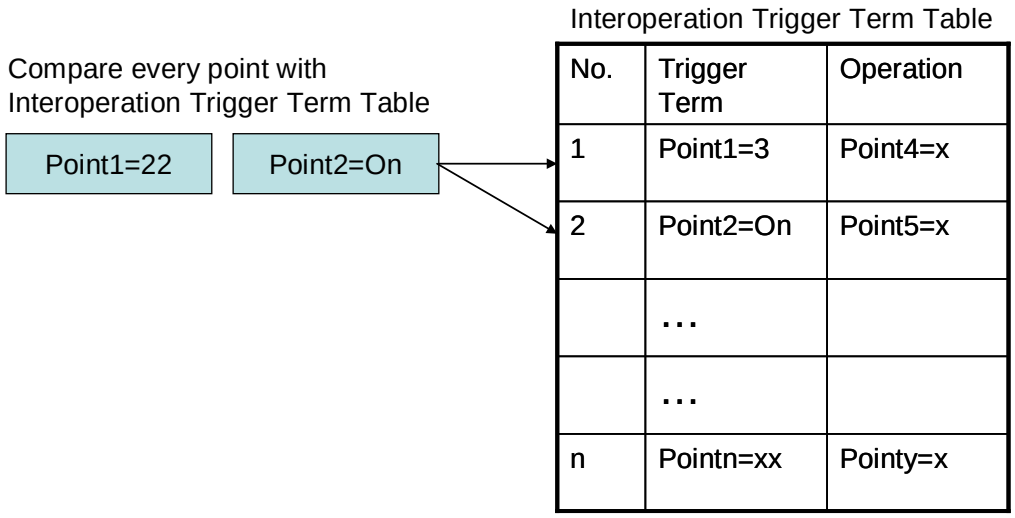


Figure 4.13 Traditional polling method for interoperation trigger term

The new design is realized by cooperation between Data Point Object and Interoperation Object as Figure 4.14. It adopts an event-driven method. Every Data Point Object has an "*interoperation-term*" property and "*InteroperationEvent*" event. When an Interoperation Object is defined, it will set the "*interoperation-term*" property of the Data Point Object by "*SetInteroperationTerm(term)*" invocation as Figure 4.15. Only when the "*interoperation-term*" of the Data Point Object meets, it

will fire an event named "*InteroperationEvent*". The Interoperation Object will capture this event and check the corresponding interlock and make the desired action or just store the value and wait for other interoperation conditions to meet (when the trigger term is composite term). This will decrease greatly the work load of the IBmanager.

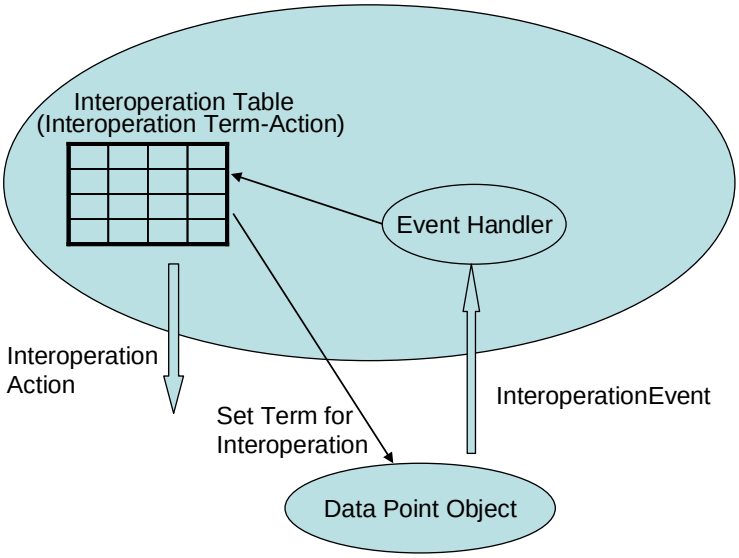


Figure 4.14 Interoperation Object Model

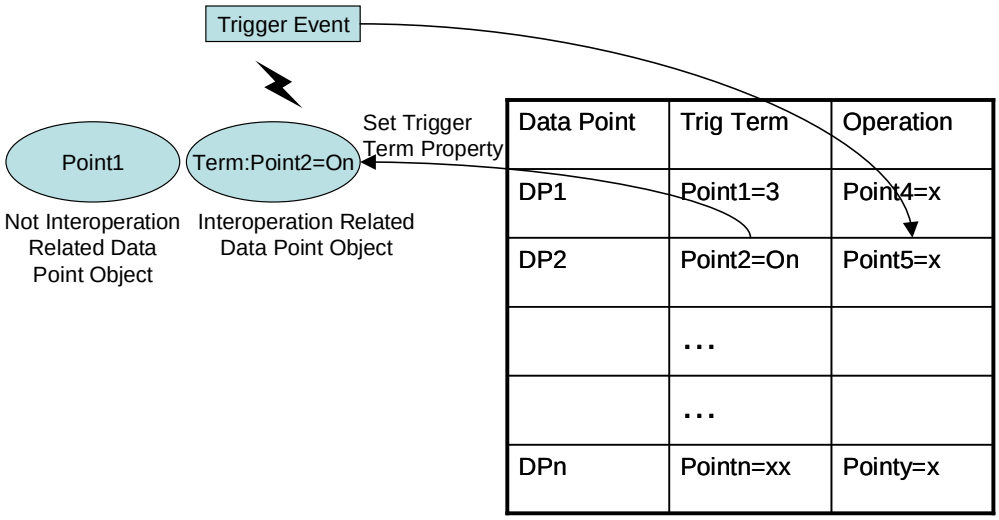


Table in Interoperation Object

Figure 4.15 Event-driven method for interoperation trigger

The interoperation trigger term table will be saved in a database, but it is kept in memory to facilitate the comparison. In VB, collection is used to maintain this table, the collection can automatically map the trigger-terms to the actions. The collection is kept in memory, no need to query to database frequently.

4.7 Database Agent Object

Database is the important part of the IBmanager to store system configuration information and historical data. Every BAS to be integrated has its own database. These databases are distributed on other computers with totally different architectures, sometimes on different locations geographically. So the IBmanager need exchange data with the remote heterogeneous databases. Section 3.4 discussed the communication technologies on local database and remote database. However, when the communication objectives come to several databases, the queries will be sent out to a few databases, engineers/developers need to deal with interfaces to a few databases. These will increase the difficulty on system operations. It is tougher especially when across-databases query, for example, query the power consumption among various buildings located at different cities.

In order to facilitate the cross-databases query and provide a common query interface for users, a Database Agent Object and several local databases are designed.

4.7.1 Database Agent Object Model

The Database Agent Object model is designed as Figure 4.16. There is a Historical Points Table maintained in the Database Agent Object, which indicates the data points to be recorded locally. This Historical Points Table is saved in the central database and been loaded to Database Agent Object on initialization. On initialization, the Database Agent Object will set the "*isHistoricalPoint*" property of the corresponding Data Point Object to "*True*" according to the Historical Points Table. When this property is set "*True*", the Data Point Object will fire "*Historical-ValueUpdate*" event when the Data Point Object has value-update beyond the deadband (if deadband is set). With the events and its input parameters, the information of historical data points is recorded into the historical database. The data points which are not historical data points will not fire this kind of events, their values will not be recorded into database.

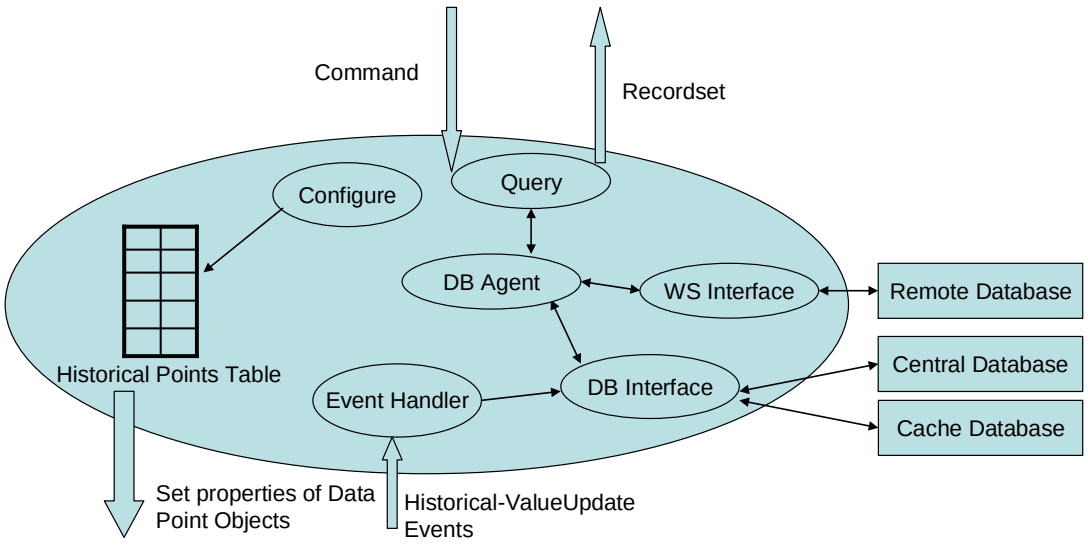


Figure 4.16 Database Agent Object model

4.7.2 Local Central Database

A local central database is installed to store the historical data with the defined format. The data in this database has two sources. One is the historical data of the local historical data points defined in the IBmanager. Another is the historical data retrieved from remote historical databases on the other BASs. User can manually download the historical data from remote databases to local central database by some transfer methods provided. In some transfer methods, user can browse the historical data points and decide which data point will be selected to transfer. The data imported of every data point will be stored as a standalone table in the central database, whose table name is named related to the name of the original data point. So the original remote database structure will not be kept in the local central database. After this operation, all the queries related to this data point will be forwarded and executed in the local central database first. However, the imported/downloaded historical data will not be updated automatically and continually, so users must keep in mind the data may be not the latest. These manually-downloaded data stored in local central database may accelerate the speed of query process to remote historical data together with the local cache database discussed as below.

4.7.3 Local Cache Database

Even with narrowing the query conditions and data compressing, the queries to remote database are time-waste task, especially frequent query. Generally there may be much repetitive data between the responses of different historical data queries. So

it is necessary to cache the return data to facilitate the query process. In order to achieve this, a **local cache database** is used to cache data from remote database as Figure 4.17. This cache database constructs and maintains one table for every cached data point, regardless which remote database and table it is from. Only historical data acquired by consecutive time query is cached, that is, the query “select A.Dp1.Value from DB1.Table as A within #Time1# and #Time2#” will be cached, the query “select A.Dp1.Value from DB1.Table as A where A.Dp1.Value= B.Dp1.Value” will not be cached since the data may not be consecutive in the original database.

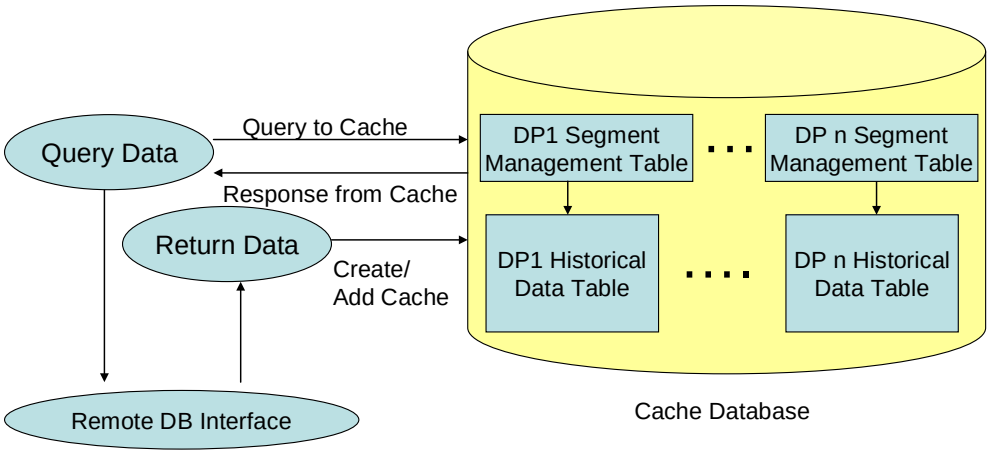


Figure 4.17 Caching remote historical data

Cached Data Segmentation and Management

Cached data may be obtained from return data of many times’ queries, the data from different queries may be not consecutive in the original database. So the cached data is divided into a few segments within which the records are consecutive originally. All these segments belong to one data point are saved in the same table. In order to mark and manage these segments, Segmentation Management Table for

cached data point is defined to maintain the data segments for every data point as Figure 4.17. The segments bounder is decided by the start and end time of the queries, regardless the timestamps of original data are consecutive or not. That means, even if the original data is not time-consecutive, they are in the same segment if they return in a query with a sole time segment condition.

When a query has data return, if the related data point has not been cached with historical data, a new cache historical data table will be created. If the data point has been cached before, the Database Agent Object will compare the start and end time of every segment, and decide whether adding some new data to the cache as Figure 4.18. So the cache will keep all the maximum historical data about the queried data point. If return data for a new query covers the discontinued time segment, the new data will be inserted into the corresponding time segment and amend the data gap. After amending, the discontinued segments may be converged to become one new consecutive segment.

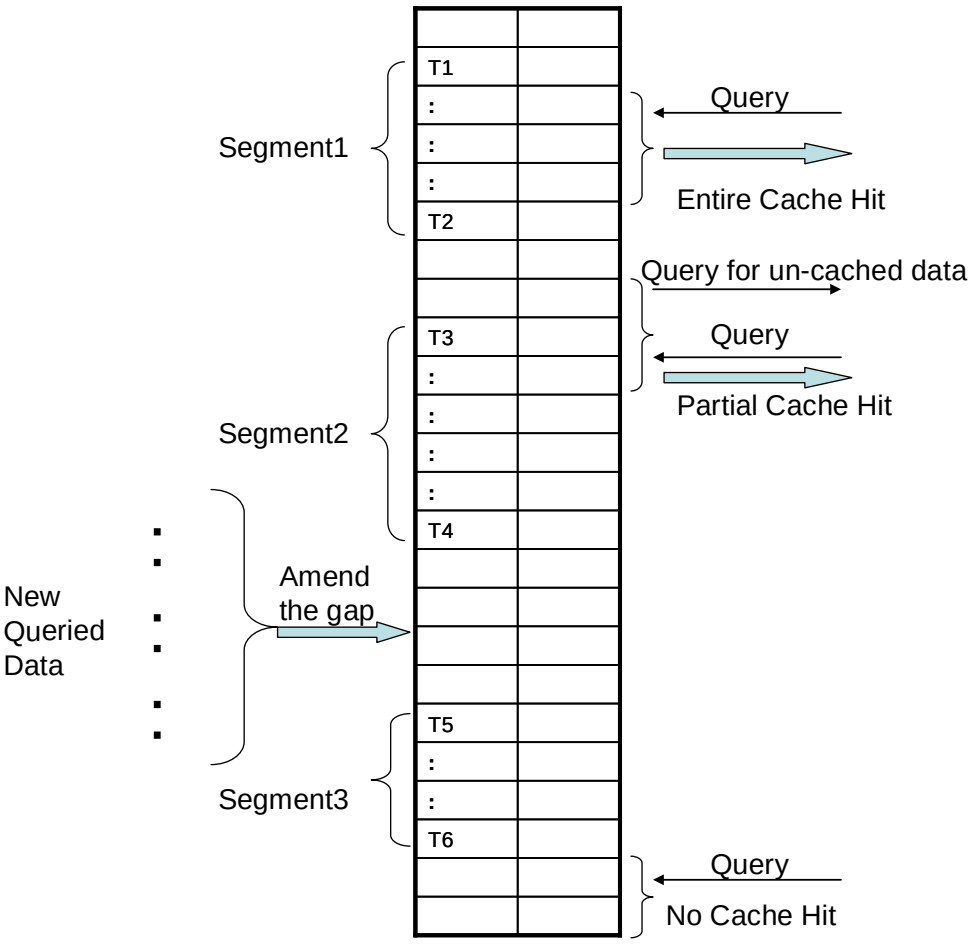


Figure 4.18 Principle of caching and querying data

Entire/Partial Cache Hit

When Database Agent Object queries the cached table, it will compare the start and end time of all segments in the cached table with the query condition as Figure 4.18. If the start and end time of any queried segment is within the duration of the cached table, an entire cache hit is achieved. If only one of start time or end time of any queried segment is located within the duration of the cached table, a partial cache hit is achieved. Otherwise, cache fit fails. In partial cache hit case, the time term in the query will be modified and narrowed to the un-cached part. The query for un-cached

part is then directed to the original remote database. This method increases greatly the chance of cache hit, and decreases the network traffic.

4.7.4 The Common Database Access Interface

Database Agent Object is designed responsible for the database access and maintenance. The information in database is mainly configuration setting and historical data in local or remote. User can access the data whether the queried database is local or remote database although the query methods may be totally different. The local database is constructed and maintained by the IBmanager, the IBmanager has absolutely knowledge and permission to query and operate local database. So it is easy to access the local database. The access to remote database is totally different. The remote databases are constructed and maintained by various providers with different technologies.

However, a common database query method is achieved by Database Agent, user can query the database no matter that the database is local or remote just like the database is a local database as Figure 4.19. Thus the operations to these heterogeneous databases can be unified. All the queries to databases are received by this agent, and then are distributed by this agent to target databases. The compositive query related to several databases will be decomposed to several queries to several single databases by the agent. The return data queried from these physical databases will be recomposed in the agent and then sent to the requester.

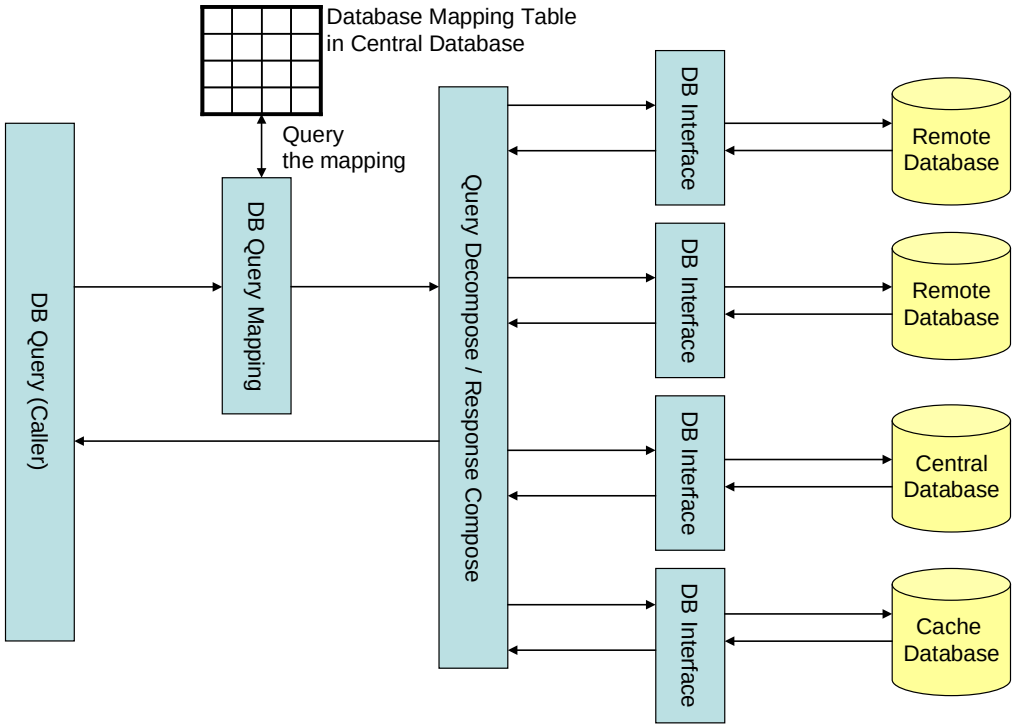


Figure 4.19 Architecture of Database Agent

Map Abstract DB Query to Queries to Physical DB

Users can issue out simple abstract query to the agent which only relates to the data point name, not assign any database name, although the query may involve to different databases. In order to finish the query, the query needs to be redirected to specific physical databases. For example, user can issue out a query “select Dp1.Value, Dp2.Value where Dp1.Value =Dp2.Value”, in this query, user needn’t assign which database and which table Dp1, Dp2, Dp3 are in, they just simply think they are in the same database. In the “DB Query Mapping”, the data points are mapped to the real physical databases and tables as Figure 4.19. For example, the query above will be mapped as “select A.Dp1.Value, B.Dp2.Value from DB1.Table as A, DB2.Table as B where A.Dp1.Value =B.Dp2.Value”, in this stage, the data points are specific to

specific database and specific table.

A mapping table named as “Database Mapping Table” is created in local central database to maintain the mapping relationship of data points with the corresponding historical database as Table-3. With the help of this table, the agent accomplishes the mapping and redirecting. This mapping table is firstly constructed by scanning the historical data points and records in the remote and local databases, some measurements are taken to make this mapping table updated. In one situation, when a new historical recording for a data point has been added in the remote database, user need add this record entry manually or rescan this database since the IBmanager have no idea about this change automatically. In another situation, when new historical recording for a data point is increased in the central database, or data records from remote database are imported to the central database, or a new record data for a data point is cached in local cache database, the new entries will be added to the mapping table automatically.

Table-3 Example of Database Mapping Table

Data Point Name	Host Name	DB Name	Table Name	Imported	Cached
DP1	Rd-Lab	Historian	Floor1	Yes	Yes
DP2	Station1	HData	Electric	Yes	No
DP3	Station2	HD	Lighting	No	Yes
DP4	Server1	LD	HVAC	No	No

If one data point has local data and remote data at the same time, the local data will be accessed first. If one data point has data from central database and cache database, the query to the cache database will be executed first. If the cache hit fails, the query will be redirected to the central database. If the central database doesn't meet the query, the query will be redirected to the original database then. So the priority from high to low is: cache database, central database, remote database.

Query Decomposition/Response Composition

Since a query may be composite and refer to a few databases, no methods for across-databases query exist directly. This composite query must be decomposed to a few queries to physical databases. In order to decompose the composite query and distribute decomposed queries to various physical databases, a decomposition rule of composite query must be defined. Several rules of the decomposition include:

- The queries must be decomposed until they refer to sole database, and
- Will not continue when the query refer to sole database, and
- The query must be decomposed to a consecutive time segment.

Let's take the query above as example, the query "select A.Dp1.Value, B.Dp2.Value from DB1.Table as A, DB2.Table as B where A.Dp1.Value =B.Dp2.Value" relates to various databases. It must be decomposed before it can be executed. This query will be decomposed to "select A.Dp1.Value from DB1.Table as A", "select B.Dp2.Value from DB2.Table as B", "select Dp1.Value, Dp2.Value where

Dp1.Value =Dp2.Value". The first two queries will be forwarded to the target remote database and the return data will be cached when they are from remote databases, the last query will be executed based on the cached data in the local cache database.

4.8 Session Object

4.8.1 Session Object Model

The Session Object keeps the information of all the client connections as Figure 4.20. Session Object is the sole components interfaced to the outside. Every client connection is managed by a Session Object.

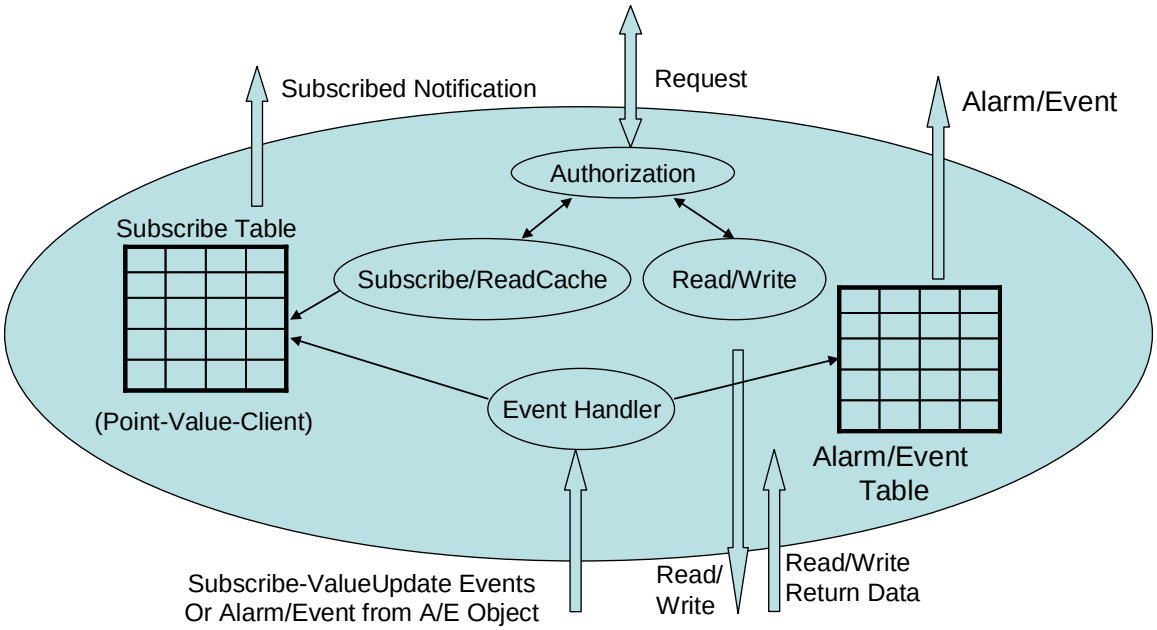


Figure 4.20 Session Object model

The Session Object is responsible to communicate with client applications. As a component of the IBmanager, it can receive and dispatch the requests from the client

after authorization. It has two methods to send data to the client applications to realize bi-directional communications. One is sending direct requests to the client applications by Peer to Peer method if the client host is a Web server as well, another is sending notification to the client according to the subscription table by “piggybacking” method.

The main communication of the Session Object and the client is subscription/notification method. The information about subscription/notification is maintained as Subscription Table. There are two methods to add data points to Subscription Table. The method one is, when one client initiates a request to a data point at the first time, the IBmanager will automatically add this data point to the Subscription Table; this is an implicit subscription. The method two is explicit “subscribe” command. After receiving the “subscribe” command, the Session Object will add the related subscribed data points to the table. This Subscription Table is maintained in memory. This table always keeps the values of subscribed data points in cache for faster access by client.

4.8.2 Communication between Session Object and Client

There are two kinds of communication between Session Object and client, one is read/write command from client to the IBmanager, the other is subscribe/notification method to keep the client updated.

i. Read/Write command from client

The client initiates “Read/Write” requests to the Session Object directly. The “Read/Write” command will be passed on to corresponding Data Point Object. The return value of the Data Point Object in “Read” command will be sent to the Subscription Table to keep the table updated if this data point is subscribed. If the “Write” command is executed successfully, the response packet (acknowledge packet) will be sent back from the Session Object. Meanwhile, if the commanded data point is in the Subscription Table and the value of this data point changes, a “*Subscribe-ValueUpdate*” event will be fired in the Data Point Object to update the cached value in the Subscription Table.

ii. Subscribe/Notification

In this method, the client subscribes the data points which it is interested in, and then the Session Object will send out the notification to the client.

The Session Object sets the “*isClientSubscribed*” property of Data Point Objects according to the Subscription Table. Once the “*isClientSubscribed*” property is set “*True*”, Data Point Object will fire “*Subscribe-ValueUpdate*” event when its value changes beyond the deadband. When the Data Point Object fires this event, event handler in the Session Object will judge whether the events are from the data points subscribed. If yes, it will synchronize the value of the corresponding data point in the Subscription Table to the new value, thus the new values will be kept in the Subscription Table to serve the client by *ReadCache* method. On the contrary, it will ignore this update event. In this kind of design, the Session Object only cares about

the events of the data points subscribed by its corresponding client. This decreases the overhead.

The Session Object monitors the Alarm/Event messages from Alarm/Event Objects as well. If an Alarm/Event message from Alarm/Event Objects is what the Session Object cares about, it will be handled and distributed to its corresponding client.

CHAPTER 5 IMPLEMENTATION ISSUES OF IBMANAGER PLATFORM

5.1 Distributed XML Driver Model

The sub-systems/devices usually distribute on different computers, even on different locations geographically. The IBmanager need communicate with these various sub-systems/devices to realize the monitoring and control functions. Traditionally these communication processes can be achieved by various specific drivers which are usually realized by some APIs invocation. Every application needs a specific driver for one protocol. From the discussion in Section 4.3.3, we know that traditional method is a complicated N*m architecture to be maintained difficultly.

In traditional drivers, the computer as a center is installed with all the drivers, connects to sub-systems/devices by various physical interfaces, including serial port, TCP/IP. All these physical interfaces need connect to the central computer directly, hardly connect across the Internet. This is another disadvantage of the traditional drivers.

In order to simplify the architecture and equip the drivers with Internet communication capability, distributed XML Driver model is designed. In the design, the communication between the IBmanager and distributed XML Drivers is conducted by XML communication technology. The XML messages may be transported on various technologies, for example, TCP, HTTP, FTP, SNMP, etc. This

provides great flexibility for users. When using HTTP as the transportation, the XML Driver is actually Web Service-based driver. Below we will elaborate the distributed XML Driver with HTTP as transportation.

5.1.1 Two Kinds of XML Driver Models

There are two kinds of implementation models of XML driver. In model one the Driver Object acts as XML Server (XML Server Driver), in model two the Driver Object acts as XML Client (XML Client Driver).

A. XML Client Driver Model

In this model, Driver Object in the IBmanager acts as XML Client, the remote counterpart – remote XML Converter acts as XML Server. The remote XML Converter is used to convert other protocols to XML/Web Service. The IBmanager as a XML Client initiates requests to the remote side. The IP address and communication port of the remote XML Converter should be set and known by the IBmanager in the configuration process. The IBmanager will communicate with the remote XML Converter by the assigned IP and communication port.

(i) Control from the IBmanager to sub-systems/devices

The XML Client Driver as a client initiates requests to the remote XML Converter when the IBmanager needs to send command to sub-systems/devices. The remote XML Converter will respond with acknowledge as Figure 5.1.

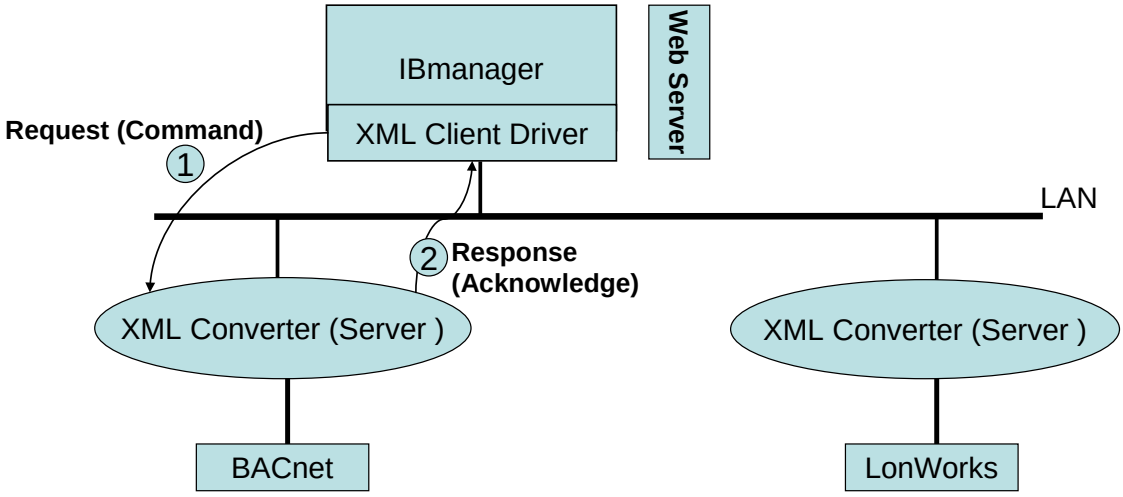


Figure 5.1 XML Client Driver model – command from IBmanager to devices

(ii) Send notification from sub-systems/devices to the IBmanager

As discussed before, XML/HTTP is stateless and based on request/response model. Only http client can initiate request to http server, the reverse operation is not supported. In this designed driver model, the remote XML Converter acts as XML Server (it is Web server as well since http is adopted as transportation technology), the computer installed with the IBmanager is equipped with Web Server as well. Thus both sides can act as Web Server, this increases the flexibility of achieving bi-directional communication greatly. The two methods discussed before, that says, Peer to Peer method and “piggybacking” technology, can be used to send notification from the remote XML Converter (Server) to the IBmanager. In the Peer to Peer method, the remote XML Converter (Server) acts as a client sends notification in its request to the Driver Object in the IBmanager directly as Figure 5.2. The Driver Object in the IBmanager acts as a Web Service provider to process the Web Service request (notification message).

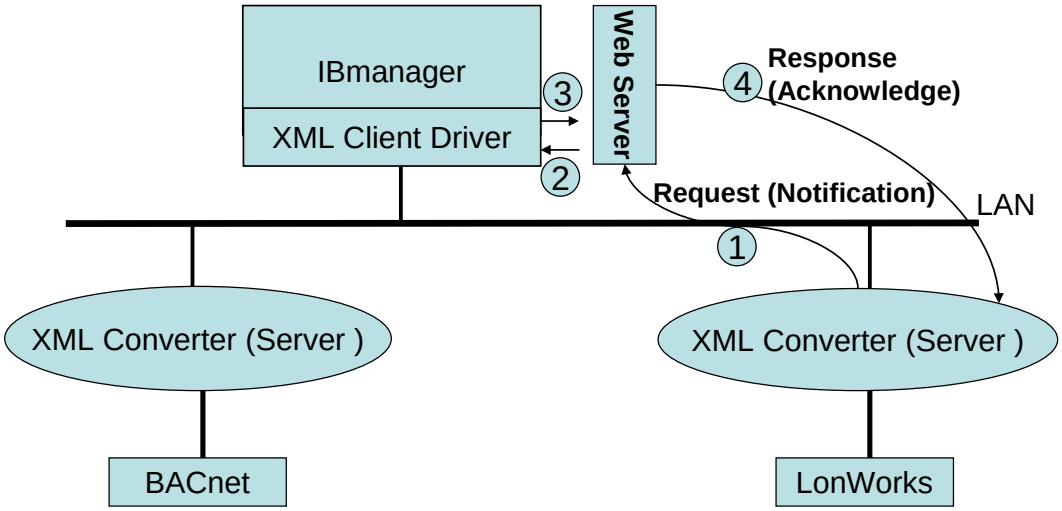


Figure 5.2 XML Client Driver model – “Peer to Peer” technology

When “piggybacking” technology is used, the notification message will be piggybacked within the response from the remote XML Converter (Server) to the IBmanager. When the IBmanager issues command to the remote XML Converter (Server) and the remote XML Converter (Server) responds to the IBmanager, the remote XML Converter (Server) will check whether it has notifications to send to the IBmanager besides the desired reply. If yes, the notification message will be sent out within the response packet as Figure 5.3. When the IBmanager issues no command to the remote XML Converter (Server), the IBmanager will send null-contented request to the remote XML Converter (Server) frequently to achieve the same function.

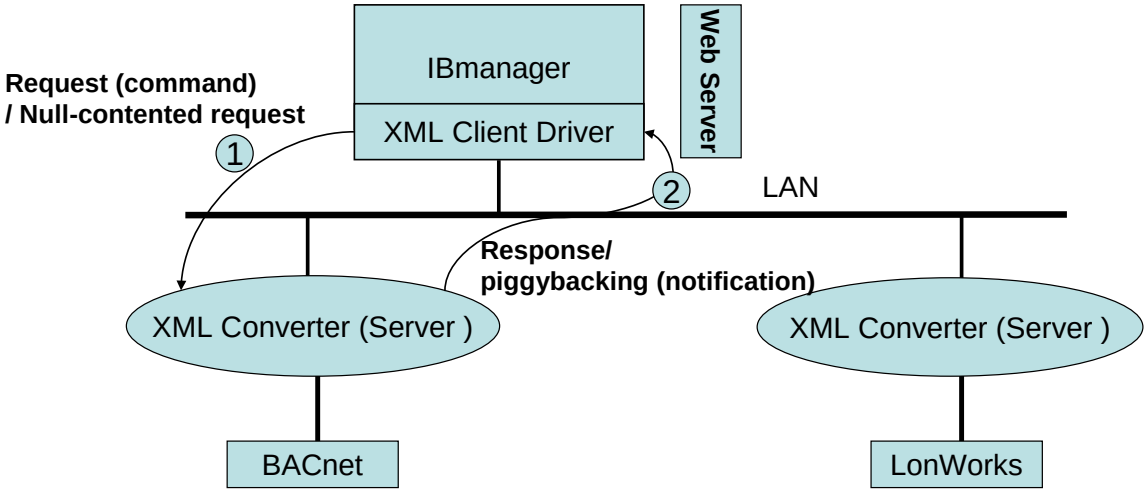


Figure 5.3 XML Client Driver model – “piggybacking” technology

2. XML Server Driver Model

In this model, Driver Object in the IBmanager acts as XML Server, the remote counterpart – the remote XML Converter acts as the XML Client. The remote XML Converter (Client) need know the IP address and communication port of the IBmanager at configuration. The remote XML Converter (Client) will connect to the IBmanager according to the assigned IP and communication port. The bi-directional communication is achieved as follows.

(i) Control from the IBmanager to sub-systems/devices

In this design, the remote XML Converter (Client) usually will not be deliberately deployed with a Web server. So the Peer to Peer technology can not be used, the “piggybacking” technology is the suitable choice. The XML Server Driver in the IBmanager will issue command to the remote XML Converter (Client) using the “piggybacking” technology. When the IBmanager receives the request issued from the

remote XML Converter (Client) and makes response, it will check whether it has something (it means command message here) to send to the remote XML Converter (Client). If yes, the command message will be sent out within the response packet as Figure 5.4. When the remote XML Converter (Client) has no request to the IBmanager, the remote XML Converter (Client) will send request with null-contented frequently to the IBmanager to achieve the same function.

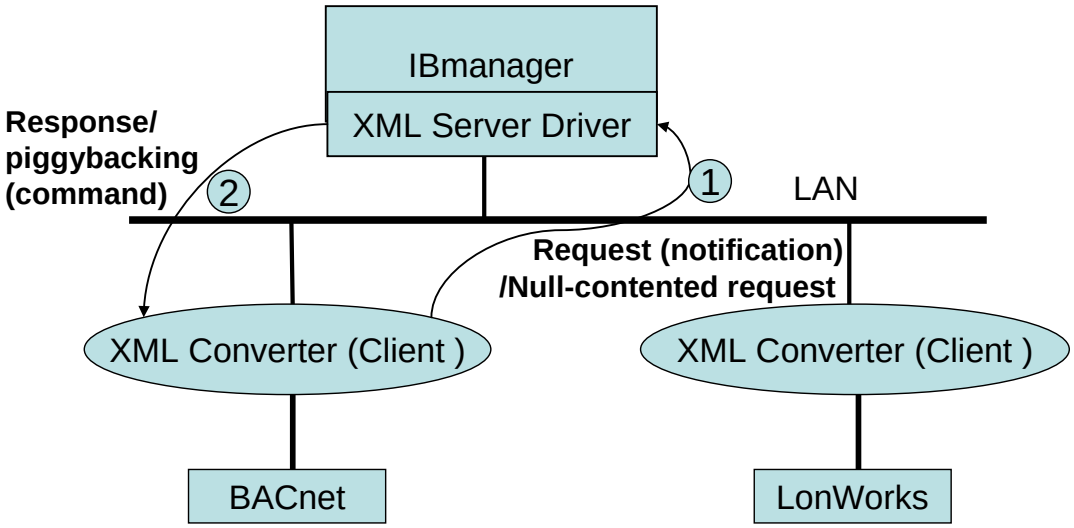


Figure 5.4 XML Server Driver model - command from IBmanager to devices

In this communication process, the computer installed with the remote XML Converter (Client) needn't host an http server (Web server).

(ii) Send notification from sub-systems/devices to the IBmanager

The remote XML Converter (Client) as a client initiates requests to the IBmanager when it needs to notify its change-of-value (COV) or alarm/event to the IBmanager as Figure 5.5.

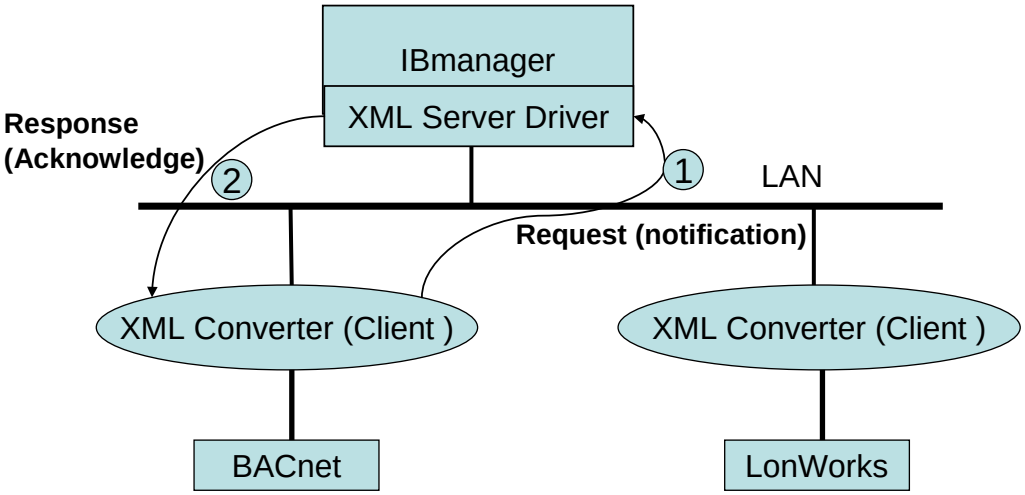


Figure 5.5 XML Server Driver model – send notification to IBmanager

3. Comparison of these two models

These two models can be compared as below:

- These both models can use the same “piggybacking” technology. Although “piggybacking” has the same time performance, the “piggybacking” contents in these two models and their influences to performance are different. The information piggybacked in the XML Server Driver model is command from the IBmanager, however it is alarm/event or COV notification in the XML Client Driver model. Command should have higher priority and be sent out immediately, however the “piggybacking” technology will result in latency. So from the view point of sending command to devices fast, XML Client Driver model is better.

- In the XML Client Driver model, Web server (if XML/HTTP is used) is deployed in the remote XML Converter side. Sometimes this is a disadvantage or blocked by firewall.

- Peer to Peer transportation is supported in the XML Client Driver model. It is usually not supported in the XML Server Driver model. The advantage of Peer to Peer method is that it will not lead to latency.

- The easiness of the configuration: In the XML Client Driver model, the IP addresses and communication ports of the remote XML Converters are configured in one location - the IBmanager. In the XML Server Driver model, the IP address and communication port of the IBmanager should be set and known to every remote XML Converter. This will increase the labor and flexibility of configuration especially when the remote XML Converters are distributed geographically.

Based on the considerations above, the XML Client Driver model is selected as our implementation model. At the same time, the Peer to Peer method is suggested because of less traffic and simplicity.

5.1.2 Reading Values & Caching Values in Advance

When the driver reads/writes data points of the field devices directly, the communication speed may be slow compared to the software process speed, sometimes the reading process even fails. In the reading/writing progress, the IBmanager will wait for the response and do no other thing until time-out if the reading is conducted in a blocked way. The slower communication and failure will influence the performance of the entire system. In order to avoid the possible slow field communication decelerates the performance of the IBmanager, a cache

mechanism for driver is designed.

In this cache mechanism, the driver will read the values of monitored data points in advance frequently. The reading scheme can be adjusted flexibly, if it has a large volume of points, the reading can be conducted in batches in advance (for example, ReadPropertyMultiple in BACnet protocol), and the interval can be adjusted accordingly. After that the reading data of the IBmanager from the driver is actually reading the cached value. Thus the reading speed will be greatly accelerated, and the system performance will be improved. Meanwhile since the interval can be set small, the latency will be kept in an unnoticed scope.

5.2 Human Machine Interface (HMI) Based on Web

5.2.1 Benefits of Web Human Machine Interface

Web technology is found almost everywhere. The following attributes go with the Internet, which are also valid for Web applications within BAS [55]:

- Global connectivity
- Global companies need global BMS information
- Easy to operate
- Access at any time, from everywhere, immediately
- Use mainstream technology
- IT-world compliance
- Slim, inexpensive clients

In addition, the Web technology facilitates and centralizes software maintenance, updates and support, offers new services and opens up new business opportunities.

The greatest advantage of Web application is ubiquitous access by B/S (Browser/Server) architecture without installing special software. In the BAS industry, Web application is a trend. Using the web browser, one can make configuration, monitor status, control actuators, receive and acknowledgment alarms/events, and view trend log of data points.

In order to achieve good compatibility and extensibility, Web HMI is constructed based on XML in this design. However, since the Web technology is based on http request-response model, manipulation of XML message and how to update real time data and A/Es timely to user is problems to be considered.

5.2.2 Web HMI Based on XML Message

The ASP (Active Service Page) files are located in the Web server of the IBmanager (the host of Web server may be standalone from the computer of the IBmanager). When the Web browser submits a request to the Web server, the Web server will parse this request and send request to invoke Web Services in the IBmanager and then get XML response from the IBmanager. The return XML message includes BAS information which the user requests. In order to show this message to user, the XML message should be transformed to html with Extensible Style Sheet Language Transformations (XSLT) file as template and displayed in the

user’s browser.

There are two methods to realize the transformation from XML message to HTML web page. Method one is to transform return XML messages to html file in the Web server, and then send it as an html file to the web browser. It can be achieved to generate a single HTML by applying a stylesheet at server side on XML document. This does not enforce client browser support for XML. The asp code is like below [75]:

```
Dim myXML,myXSL
myXML=Server.CreateObject("Microsoft.XMLDOM")
myXSL=Server.CreateObject("Microsoft.XMLDOM")
'Fill myXML with one of the methods mentioned above
'Load a stylesheet
myXSL.load "xsl_path"
strHTML = mydoc.transformNode(myXSL.documentElement)
Response.write strHTML
```

Another method is to generate a full XML document and sending it to the client as is, a stylesheet reference is included in the XML itself and browser will apply that stylesheet to the XML document to compose html file. This requires that the client browser supports XML. The asp code is like below [75]:

```
Dim mydoc,myelement
Set mydoc=Server.CreateObject("Microsoft.XMLDOM")
```

```
mydoc.load("xml_file_path")
Response.ContentType = "text/xml"
Response.write mydoc.xml
Or
Response.write "xml_as_string"
```

Here ContentType is set to text/xml so that an XML-aware client browser knows what to do with it. Raw XML can be sent in the form of string as well.

The latter method is chosen in the middleware development in this study. In this method, XML response return from the IBmanager is forwarded to the browser by the Web server. The response XML message includes a URL (Unified Resource Location) of the XSLT file. The browser then gets XSLT file and converts the XML message into an html file. If an XSLT file was downloaded before and not updated, the browser will use the XSLT file in cache. Because XSLT files do not change frequently, only the contents of the XML message needs to be updated frequently. This will decrease the amount of traffic greatly as compared with the first method [6]. The work procedure is shown as Figure 5.6.

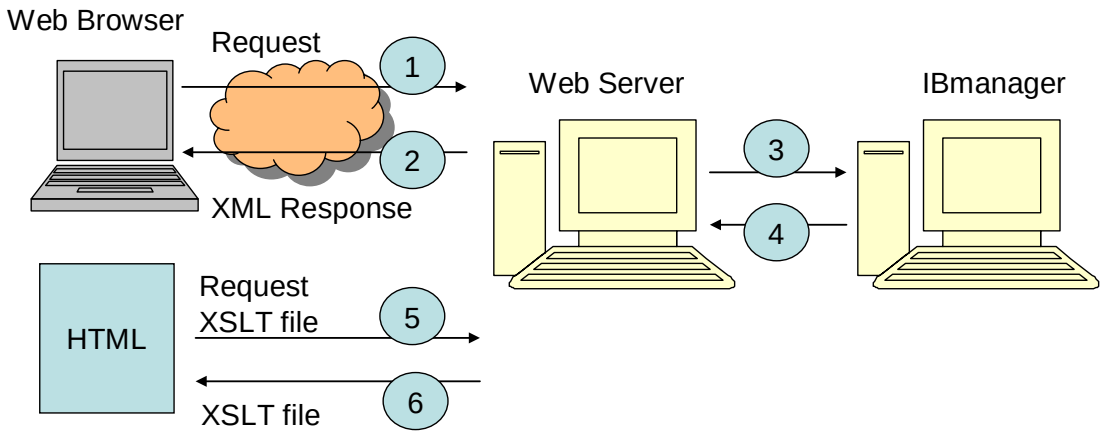


Figure 5.6 Procedure of transforming XML message to HTML web page

5.2.3 Review of Web Client Technologies Concerned

About the automatic update of the information in the user interfaces, including how the notification (including COV and alarm/event) in the IBmanager are sent to the client, how the web page interacts with user. These problems can be addressed by a few technologies. With these technologies, the notification information can be delivered to web client instantly. Below several options are discussed and provided to achieve this objective.

1. Webpage refresh method

By adding tag <META HTTP-EQUIV=REFRESH CONTENT= "time; URL=url" > in the web page, a browser refreshes the page according to the preset time frequently. This tag also can be sent by ASP program from server, such as “Response.Write <META HTTP-EQUIV=REFRESH CONTENT= "time; URL=url" >”. However, this refresh will refresh the entire webpage and lead to heavy

traffic and low performance, especially when loading large-size pictures. In order to decrease traffic, one should limit the items to be refreshed, such as refreshing only one ASP URL to finish the request. The web page technologies such as IFrame can be employed to restrict the items to be refreshed.

Using an IFRAME in conjunction with a script on your web server or a database of static HTML files is an easy option available. The IFRAME has been part of the HTML specification since version 4 and is supported in nearly all modern browsers. For example, code <P ALIGN=center><IFRAME SRC="foo.html" WIDTH=300 HEIGHT=100></IFRAME></P> will restrain content to foo.html in this Iframe. If foo.html includes the code to refresh automatically, it will only refresh the Iframe and will not affect the contents outside of the Iframe. On the server, you can use your scripting language of choice to process page requests made to the IFRAME [76].

However, not all browsers support IFRAME. And the IFRAME will increase the complexity of web pages when many IFRAMEs are used.

2. ActiveX control/Java Applet

ActiveX control or Applet are applications embedded in web page, they can refresh themselves and “pull” data from server periodically, or they create and keep a socket connection and listen (as a server) the messages coming from server, such as A/E message. By embedding an ActiveX component, a Java applet into your web page, one is capable of making HTTP requests and interacting with the client-side

JavaScript code.

The disadvantages are:

- Using an embedded object for remote scripting requires the end-user to install additional proprietary software;
- ActiveX does only work on the Windows platform, and Java can cause some difficulties with some versions of Internet Explorer.

Unless you're developing in an environment where browser homogeneity can be assumed, these technologies may not be a good choice for updating information at client-side [76].

3. XMLHTTP

Short for Extensible Markup Language Hypertext Transfer Protocol, XMLHTTP is a set of APIs that enables XML, HTML or binary data to be transmitted to and from Web servers over the Internet using HTTP. An advantage of XMLHTTP is that when files that are ASPs or CGI (common gateway interface) programs are queried from the server, the XMLHTTP object continuously queries the server transparently to retrieve the latest information without the user having to repeatedly refresh the browser. XMLHTTP enables streamed content through DHMTL rather than ActiveX controls or Java applets.

With the support of the XMLHttpRequest object, developers can [77]:

- Update a web page with new data without reloading the page

- Request data from a server after the page has loaded
- Receive data from a server after the page has loaded
- Send data to a server in the background
- The XMLHttpRequest object is supported in all modern browsers.

The request can be handled asynchronously. This means that the script continues to run after the send() method of XMLHTTP, without waiting for a response from the server. The onreadystatechange event of XMLHTTP complicates the code. But it is the safest way if one wants to prevent the code from stopping when no response is got from the server [77]. Below is some sample code for XMLHTTP.

```
<script type="text/javascript">
var xmlhttp;
function loadXMLDoc(url)
{
xmlhttp=null;
if (window.XMLHttpRequest)
   {// code for all new browsers
   xmlhttp=new XMLHttpRequest();
   }
else if (window.ActiveXObject)
   {// code for IE5 and IE6
   xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
   }
if (xmlhttp!=null)
   {
   xmlhttp.onreadystatechange=state_Change;
   xmlhttp.open("GET",url,true);
   xmlhttp.send(null);
```

```
    }
else
    {
    alert("Your browser does not support XMLHTTP.");
    }
}function state_Change()
{
if (xmlhttp.readyState==4)
    {// 4 = "loaded"
    if (xmlhttp.status==200)
        {// 200 = OK
        // code to write the response to components of web page
        }
    else
        {
        alert("Problem retrieving XML data");
        }
    }
}
</script>
```

The XMLHTTP object queries the server transparently to retrieve the latest information. The response can be retrieved by XMLHTTP and be processed by script components. Thus it realizes the page update without refresh the entire page.

4. Ajax

Ajax (also known as AJAX), shorthand for "Asynchronous JavaScript and XML", is a web development technique for creating interactive Web applications. The intent is to make web pages feel more responsive by exchanging small amounts of data with

the server behind the scenes, so that the entire web page does not have to be reloaded each time the user requests a change. This is intended to increase the web page's interactivity, speed, and usability [78].

Ajax isn’t a technology, it is really several technologies, each flourishing in its own right, coming together in powerful new ways. Ajax incorporates [79]:

- standards-based presentation using XHTML and CSS;
- dynamic display and interaction using the Document Object Model;
- data interchange and manipulation using XML and XSLT;
- asynchronous data retrieval using XMLHttpRequest (also known as 'remote scripting');
- and JavaScript binding everything together.

The classic Web application model works like this: Most user actions in the interface trigger an HTTP request back to a web server. The server does some processing — retrieving data, crunching numbers, talking to various legacy systems — and then returns an HTML page to the client. While the server is doing its thing, what’s the user doing is waiting. And at every step in a task, the user waits some more. This approach makes a lot of technical sense, but it doesn’t make for a great user experience [79].

An Ajax application eliminates the start-stop-start-stop nature of interaction on the Web by introducing an intermediary — an Ajax engine — between the user and

the server. Instead of loading a webpage, at the start of the session, the browser loads an Ajax engine — written in JavaScript and usually tucked away in a hidden frame. This engine is responsible for both rendering the interface the user sees and communicating with the server on the user's behalf. The Ajax engine allows the user's interaction with the application to happen asynchronously — independent of communication with the server. So the user is never staring at a blank browser window and an hourglass icon, waiting around for the server to do something [79].

4. Conclusive Remarks

Although these several methods can keep the HMI updating data and receiving alarms/events notification timely without reloading the entire web pages. Users can select one or several according to their development environments, even just their habits.

XMLHTTP and Ajax can greatly promote user interaction experience and keep the value update with low traffic. It is a good choice for web applications. Although XMLHTTP and Ajax are originally presented for prompting user interaction experience, they can be uses to request and update value of data points automatically as well.

5.3 Concurrent Operation

5.3.1 Concurrent Command

There exists situation in which several clients to control data points simultaneously. These may lead to abnormal operation and over-frequent actions. In order to keep the system operate correctly in this situation, a command priority mechanism is defined. Every login user or client connection has assigned a specific privilege, for example, from 1 to 16. Value 1 is the topmost privilege. When multi-clients send commands to the server simultaneously, all the commands will filled into the corresponding elements in a priority array according to the assigned priority as Figure 5.7. The final execution will depend on the element with highest priority. The command with highest privilege will be executed.

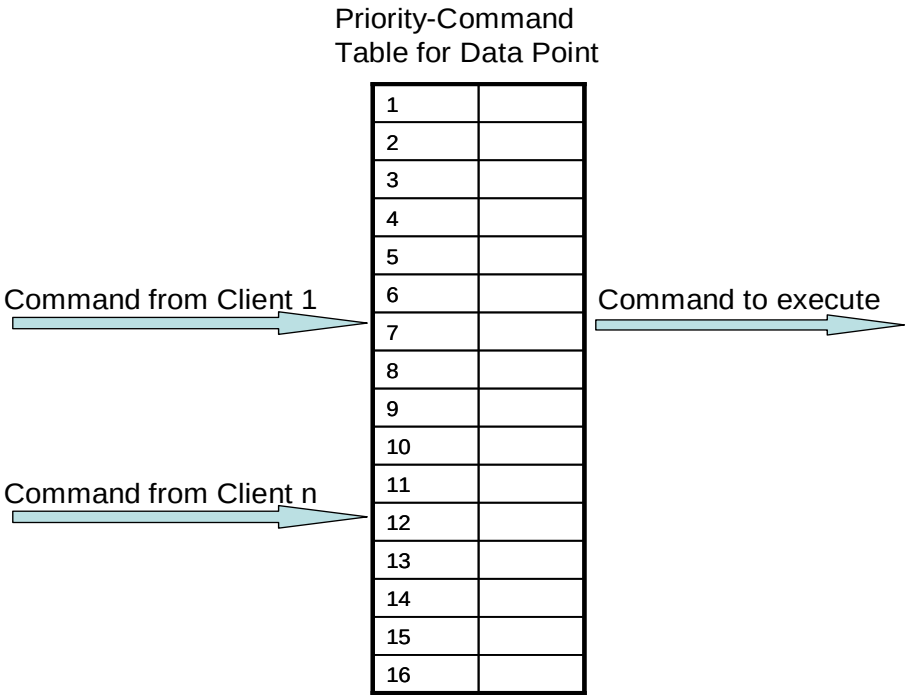


Figure 5.7 Command based on priority mechanism

5.3.2 Anti-fluctuation Operation

Frequent startup-stops may lead to damage for some facilities, for example, chillers. In order to keep the devices from fluctuate needlessly, after a command is executed, the executed command will promoted to a priority higher than the priority of the sequence control logic or other command source and stay on the higher priority level for an interval. In this interval, commands from the sequence control logic or other command source will not execute since the just-executed command keeps on a higher priority. After this interval, the just-executed command will return to its original priority. Other commands with higher priority will be effective. This process is shown as Figure 5.8.

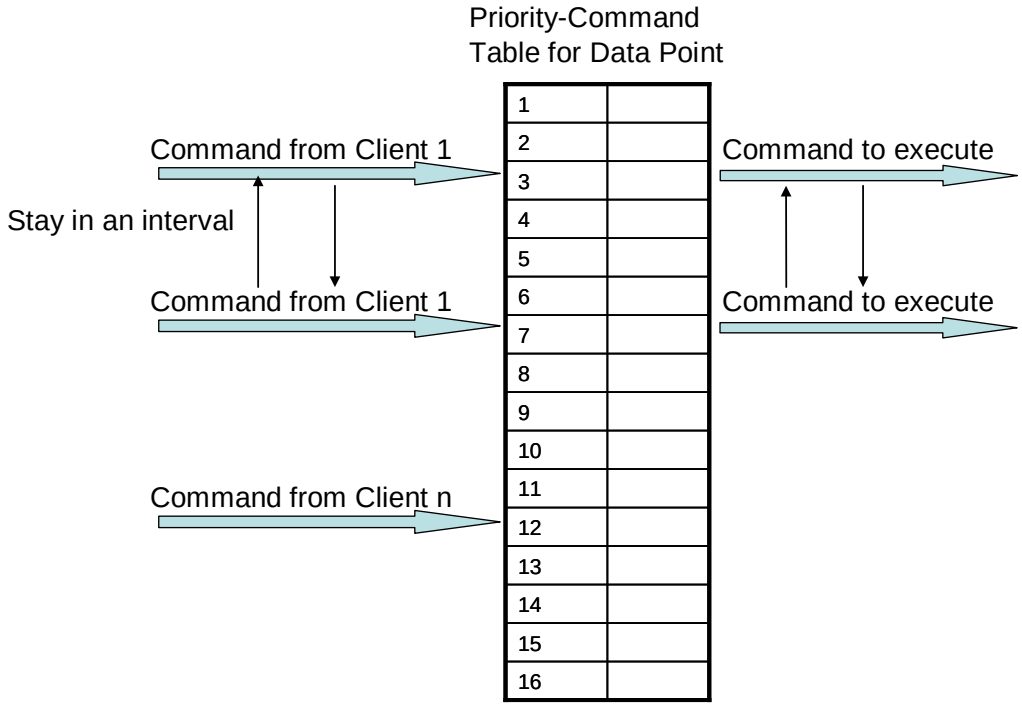


Figure 5.8 Anti-fluctuation mechanism

5.4 Redundancy and Fault-tolerant

For the reliability of the system, the IBmanager is configured to be redundant. Let's take two IBmanager installations as example. One server acts as primary server, another server acts as backup server. The server running as primary communicates with the sub-systems/devices. If the primary server fails, the backup server becomes the primary and takes over communications with the sub-systems/devices. For example, if Server A is running as the primary server and fails, the arbitration software detects this and switches Server B from running as backup to running as primary.

When the primary and backup servers are running in redundant mode, all database changes on the primary server are sent to the backup server to keep the two servers to have consistent data. The date and time on the primary and backup servers need to be synchronized to ensure that all dates and/or times associated with events in the database are consistent between servers. In the design, the primary server is used as the time source and the backup server is configured to synchronize with the primary server. This process is illustrated as Figure 5.9.

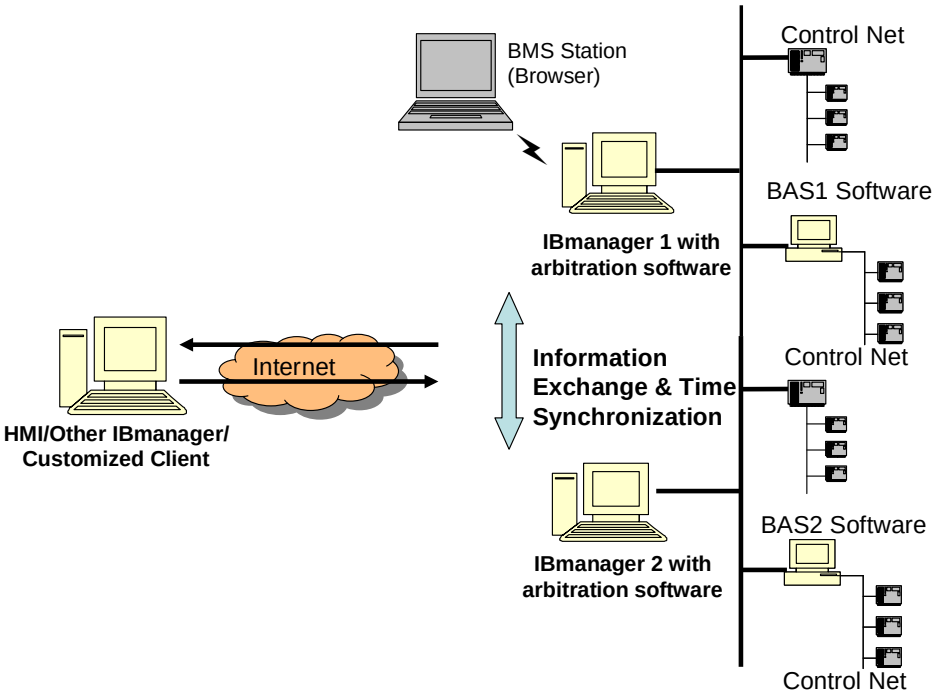


Figure 5.9 Redundant IBmanager installations

Redundant arbitration is the task of deciding which of the servers will run as the primary server and which will run as the backup server. The arbitration based on software is called software arbitration. In some software arbitration, software running on the primary and backup servers provides the arbitration. Each server polls the other (via the network) so that it knows if the other server has failed [80]. However, if a server loses connection to another server, how can it be judged which server fails? Either one is possible to have failed. A new arbitration method is designed for accurate judge as Figure 5.10. Server A and Server B will modify one data point in a device to different value frequently, for example, Server A modifies it to value 1, Server B modifies it to value 2. Every server will judge whether the other server has modified the value and therefore been working normally or failed. For example, If Server B can read the value of the data point and finds the value of the data point

hasn’t be modified to value 1 frequently, then it knows Serve A or the communication to Server A has failed.

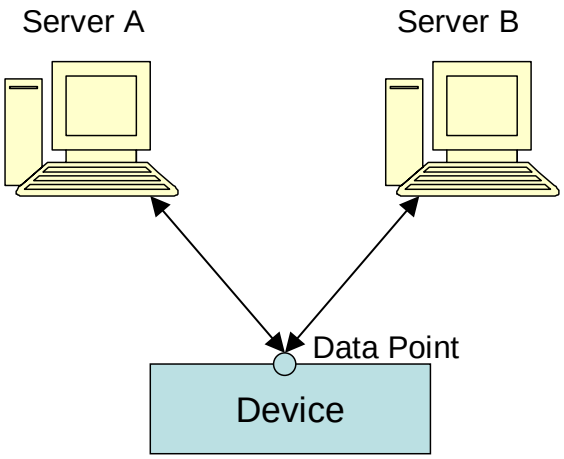


Figure 5.10 Software arbitration

5.5 Value-added Services DLL

As a powerful integration platform, the IBmanager provides a flexible way to extend its functions – the method to add customers’ value-added services, for example, HVAC optimization application, some decision analysis application. IBmanager presents a method to invoke value-added services as configurable plug-ins which can be added/removed from the IBmanager.

Users can develop their own DLL (dynamic link library) to add functions to the IBmanager. The DLL must be compliant the specifications of the interface to the IBmanager. Some configuration work should be done in the IBmanager to let it know how to invoke these DLLs with their parameters. Every invoked DLL has a corresponsive entry in the configuration file. After adding new DLL entries in the

configuration file, the newly-added entries will be submitted to the IBmanager immediately. The related DLLs will be loaded immediately. This way provides a flexible method for uses who want to add some their own functions to the IBmanager.

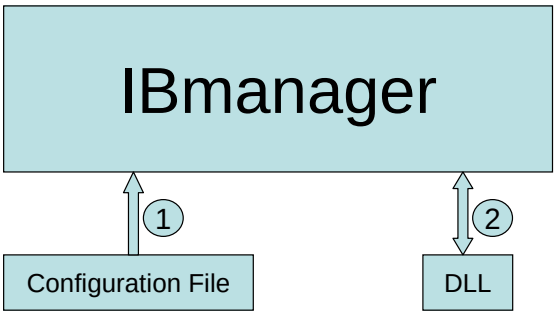


Figure 5.11 Invocation of third-party DLLs

In an practical project – ICC project which will be introduced later, the IBmanager need invoke Matlab DLL to achieve some complicated HVAC optimization calculation. In the configure information, the functions invoked and their input parameters and output parameters are assigned, the IBmanager then knows how to invoke it.

The traditional method is to hardcode the optimization calculation in the platform software. If the user has some new ideas about the calculation, the entire platform must be compiled again even a small modification.

CHAPTER 6 TEST FACILITIES AND METHODS

6.1 The Intelligent Building Lab

The Intelligent Building Lab of The Hong Kong Polytechnic University (PolyU) was founded and constructed by Professor Shengwei Wang in 2002. The IB Laboratory provides the learning/teaching test and demo facilities for postgraduate and undergraduate subjects on Intelligent Building, the test facilities for R&D on Intelligent Building technologies, as well as the test facilities for postgraduate and undergraduate student research projects.

6.1.1 The Overall Architecture of the IB Lab

The laboratory facilities include: a comprehensive IB system of Honeywell, a full scale BMS of Johnson Controls, LonWorks control networks, a home automation (X-10) system, building emulators, BACnet control networks, etc. as Figure 6.1 and Figure 6.2. All these different products provide a good chance for testing the integration platform.

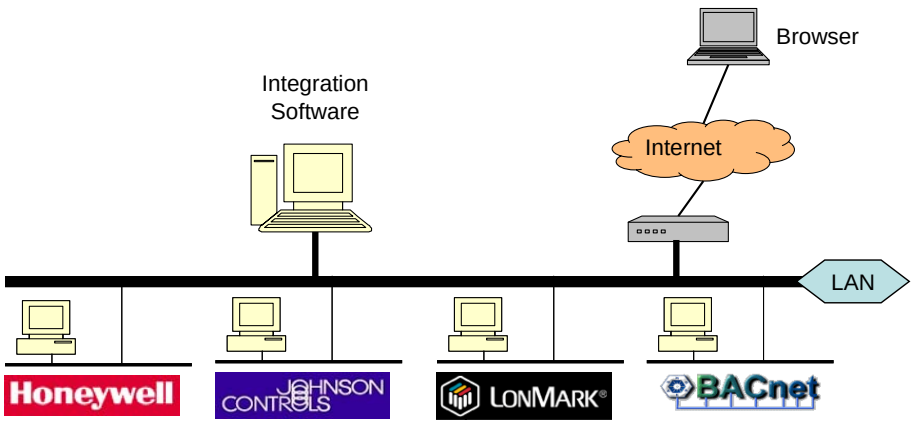


Figure 6.1 Connected products from various manufacturers in PolyU IB Lab

Figure 6.2 One corner of the IB Lab

6.1.2 Honeywell Products

The products from Honeywell can support BACnet, OPC, LonWorks, DDE interfaces. The installation in the lab includes the subsystems as below:

- HVAC control
- Security
- Digital CCTV
- Life Safety
- Lighting Control
- Power Monitoring

All these sub-systems are connected to Ethernet as the backbone as Figure 6.3.

The management software - EBI (Enterprise Building Integrator) communicates with them via Ethernet [80].

EBI provides OPC server, BACnet Server interfaces for invocation by third-party software. In the lab tests of this study, the IBmanager will monitor and command Honeywell system by OPC Server interface of EBI software.

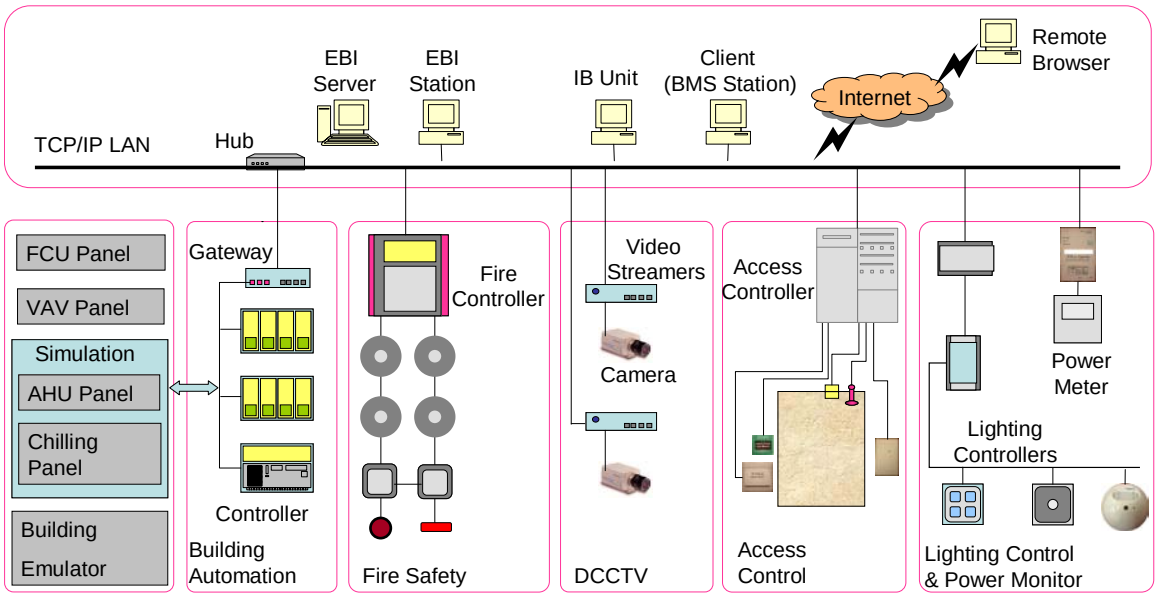


Figure 6.3 Architecture of Honeywell BAS in the PolyU IB Lab

6.1.3 Johnson Control's Products

The installation of Johnson Controls in the lab includes the sub-systems listed below. Their communication protocols including BACnet, N1 protocols [81].

- HVAC control
- Security
- Digital CCTV

- Life Safety
- Power Monitoring

All these sub-systems are connected to Ethernet as the backbone as well. The management software – M3, M5 (Metasys) communicates with them via Ethernet. They integrate the sub-systems by OPC Servers, including BACnet OPC Server, N1 OPC Server as Figure 6.4.

In the lab tests of this study, the IBmanager will monitor and command the system of Johnson Controls by OPC Server interface.

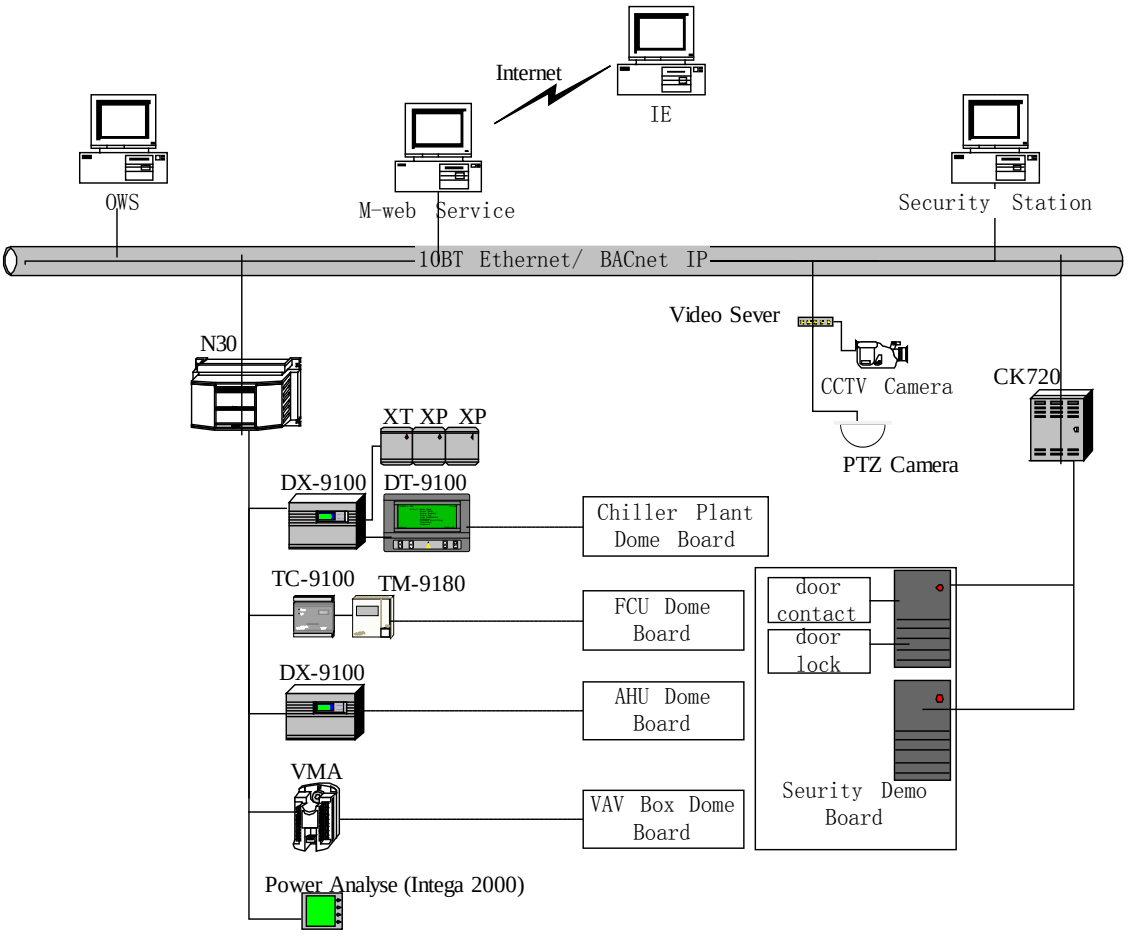


Figure 6.4 Architecture of the system of Johnson Controls

6.1.4 BACnet Products

There are also native BACnet products from (Honeywell) Alerton (Alerton has been merged into Honeywell in 2005). Their products support BACnet Class 3. The workstation software – Envision for BACtalk supports ActiveX interface instead of OPC as Figure 6.5 [82].

There are two methods can integrate (Honeywell) Alerton's BACnet products in the IB Lab. One is by ActiveX interface it supplies. In this method, a remote XML Converter can be used to convert the ActiveX interface to XML protocol. Another method is a standalone BACnet OPC Server package. This interface can read BACnet and map to OPC data items directly without accessing (Honeywell) Alerton's software.

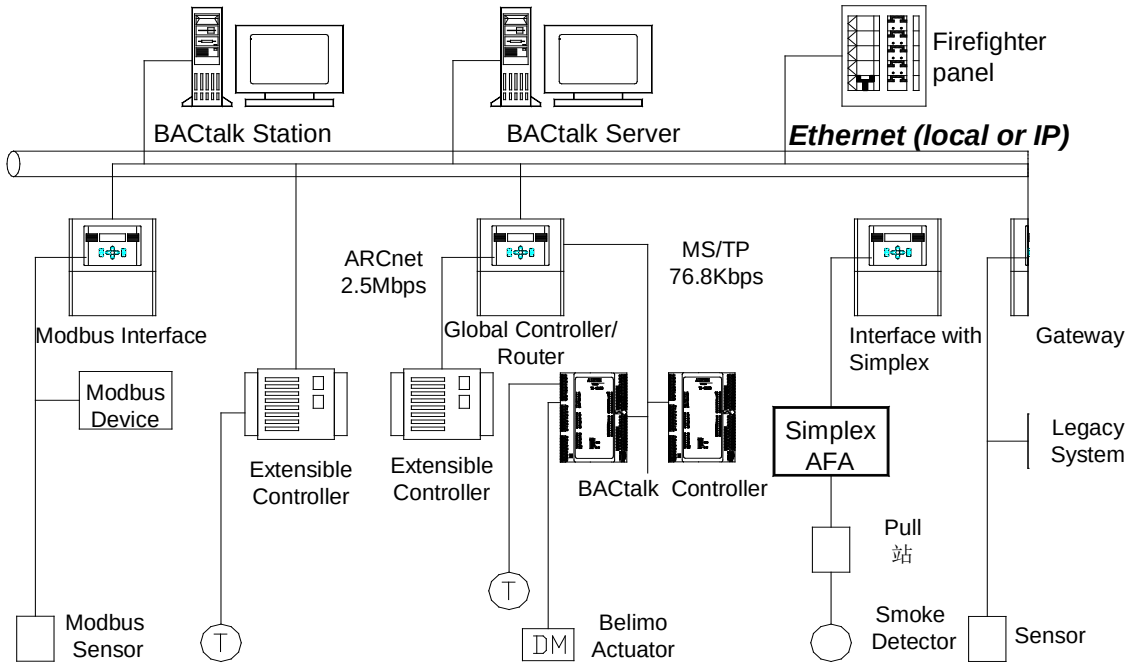


Figure 6.5 BACnet control system from (Honeywell) Alerton

6.1.5 LonWorks Products

Besides LonWorks-compliant products from Honeywell, the lab has included Building Open Network Test Kit (IBON) based on LonWorks products from Echelon in 2001 [4]. Figure 6.6 is the schematic figure of the IBON. LonMaker from Echelon provides DDE interface for other applications [83].

In the lab tests of this study, LonWorks network was accessed by DDE interface provided by commercial software – LNS DDE Server.

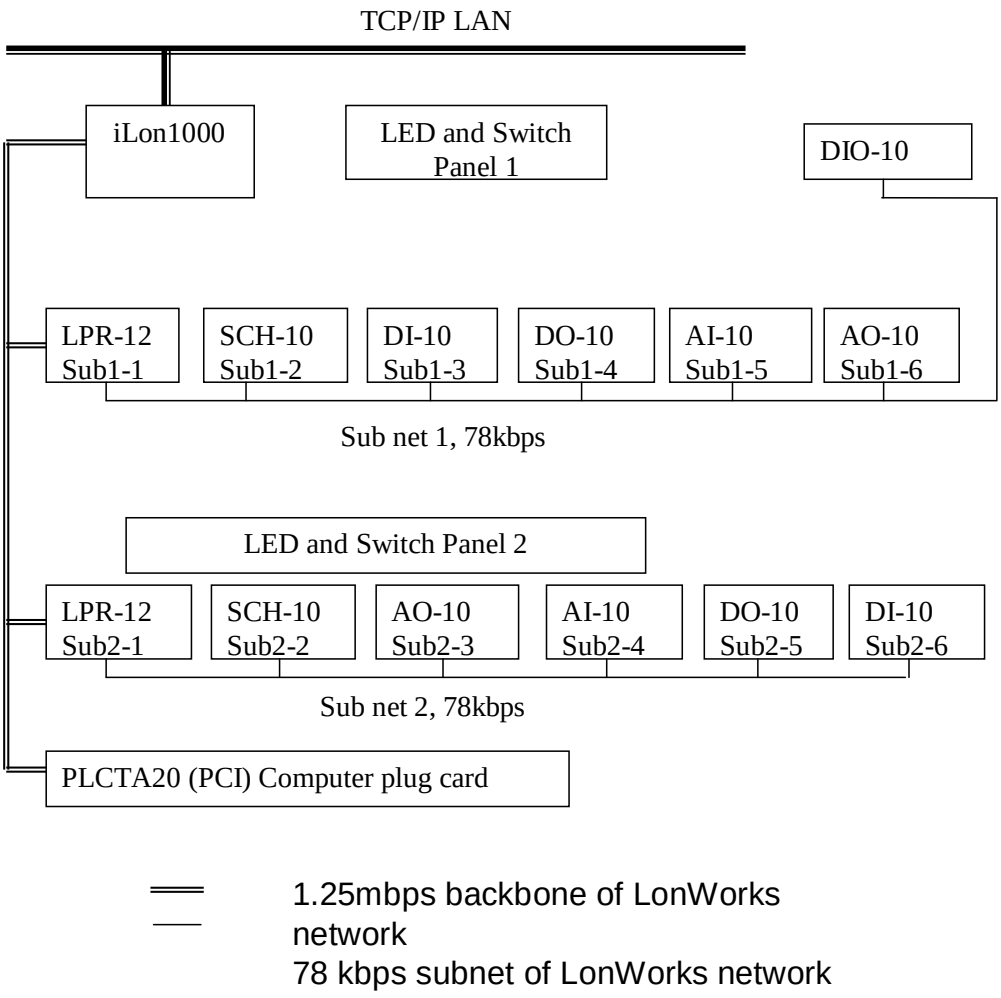


Figure 6.6 IBON Test Kit Architecture

6.2 Integration/Interoperation Test Environment

BAS products from a few manufacturers were involved in the tests of this study. The products are compliant with different protocols. The communication between different protocols may be conducted by various methods in two levels.

6.2.1 Integration at Automation/Field Level

Direct communication

Ideal BASs should be integrated in the automation/field level. The controllers which are complaint with the same protocols from different vendors can communicate each other. For example, in the lab, Johnson Control's N30 network controller supports BACnet, while (Honeywell) Alerton's system is BACnet compliant. Therefore they can be connected together directly. Honeywell's Excel5000 supports LonWorks. Echelon LonPoint series are LonWorks compliant. They can be connected together. However, there exist a lot of control devices which are complaint with different protocols in the real applications. For example, BACnet products cannot communicate with LonWorks products in the Lab. In this case expensive specific gateway is necessary, or the data exchange needs to be realized at the management level, which will be discussed later.

Integration by Gateway

Another integration method in the automation/field level is gateway. In this

method, the conversion between different communication protocols is conducted by gateway. For example, in the PolyU IB Lab, c-bus protocol in the lighting control system is connected to Honeywell EBI by converting the c-bus to LonWorks. This gateway is a product-specific, the customized development service is necessary. The communication between BACnet and LonWorks can be conducted similarly.

6.2.2 Integration at Management Level

Traditional integration of diverse drivers

In the traditional integration method of management level, sub-systems/devices are connected to the integration software by various communication drivers. All drivers for diverse interfaces are installed in the PC hosting the integration software as Figure 6.7. The disadvantage of this architecture is its N*m architecture described in Section 4.3.3.

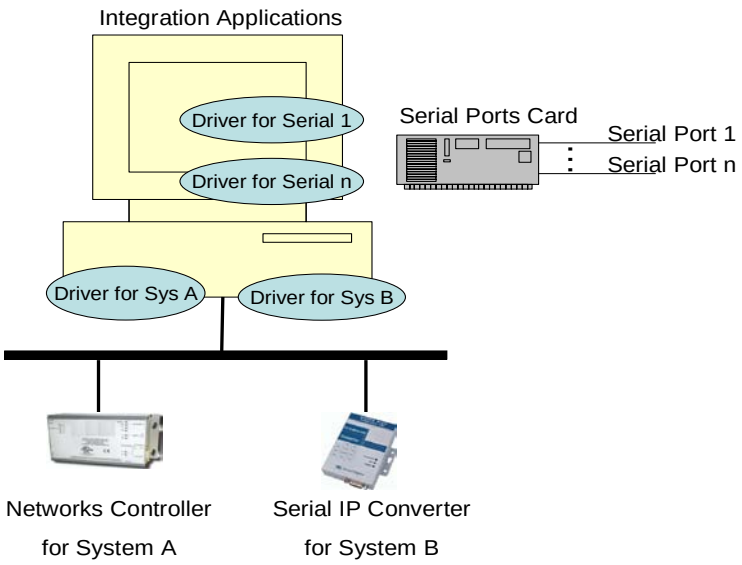


Figure 6.7 Architecture of integration by traditional driver technology

Integration based on the IBmanager

IBmanager can communicate with these various sub-systems/devices by high-level interfaces in management level or field bus protocols in field level by the unified XML Driver technology. Some products may have more than one interface to be integrated into the IBmanager. For the system of Honeywell and Johnson Controls, OPC and BACnet can be used to connect to the IBmanager. For BACnet products from (Honeywell) Alerton, ActiveX interface, BACnet and OPC can be used. For LonWorks from Echelon, LonWorks, DDE and LNS can be used. All these different interfaces are converted into XML Driver model of the IBmanager.

6.3 Communication Performance Test Method

6.3.1 Time Measurement Method

The measurement of response/process time is an important method to evaluate the performance of the middleware platform. Several time measurement methods of differing accuracy are offered by the Windows operating system [84] as Table-4.

Table-4 Several time measurement methods in Windows platform

Function	Units	Resolution
Now, Time, Timer	seconds	1 second
GetTickCount	milliseconds	approx. 10 ms
TimeGetTime	milliseconds	approx. 10 ms
QueryPerformanceCounter	milliseconds	approx. 1 ms

In our test, QueryPerformanceCounter method is used for high-resolution timings

if the computer system supports this high-resolution counter. The resolution in this case is on the order of a microsecond. Since the resolution is system-dependent, there are no standard units that it measures. You have to divide the difference by the QueryPerformanceFrequency to determine the number of seconds elapsed [84]. The sample code in VB is as below:

```
Private Declare Function QueryPerformanceCounter Lib "kernel32" (lpPerformanceCount As Currency) As Long
Private Declare Function QueryPerformanceFrequency Lib "kernel32" (lpFrequency As Currency) As Long
'DelayNum is delay count (ms)
Private Sub DelayTime (ByVal DelayNum As Long)
    Dim Ctr1, Ctr2, Freq As Currency
    Dim Count As Double
    If QueryPerformanceFrequency (Freq) Then
        QueryPerformanceCounter Ctr1
        Do
            QueryPerformanceCounter Ctr2
            Loop While (Ctr2 - Ctr1) / Freq * 1000 < DelayNum
        Else
            MsgBox "Not support high accuracy counter!"
    End If
End Sub
```

6.3.2 Measurement Method of Communication

Roundtrip time measurement

The method is to measure the time to make request and the time to receive response in the request/client side, both the request and response time is recorded in

the same communication side, their time difference is the roundtrip time of the entire communication as Figure 6.8.

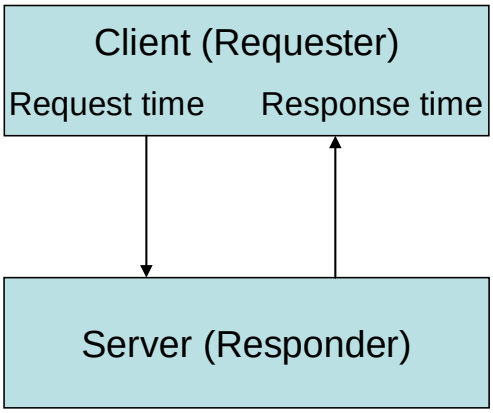


Figure 6.8 Roundtrip time measurement

One-way transportation time

In this method, the request time is transported to the server side, and then the server can calculate the time difference between the request time and the receive time to get the one-way transportation time as Figure 6.9. The time of event transportation can be measured as this. Event is the important message transportation method, the time to fire the event can be transported to the handler as parameter of the event, and then the handler can calculate the time of event transportation.

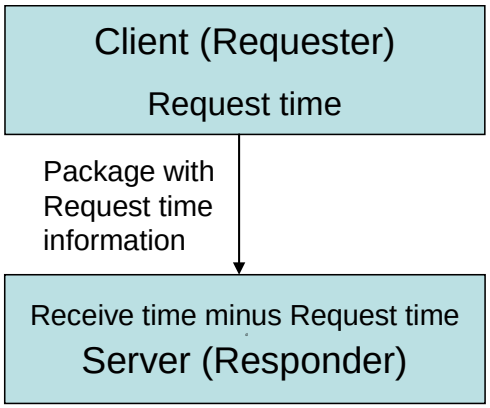


Figure 6.9 Time measurement method of one-way transportation

6.3.3 Measurement Method of Application Load

How does an application influence the computer system? How much does the IBmanager increase CPU usage and memory usage? The influence can be viewed by Windows Task Manager.

Windows Task Manager is a task manager application included with Microsoft Windows NT family of operating systems that provides detailed information about computer performance and running applications, processes and CPU usage, commit charge and memory information, network activity and statistics, logged-in users, and system services. The Task Manager can also be used to set process priorities, processor affinity, forcibly terminate processes, and shut down, restart, hibernate or log off from Windows [85].

CHAPTER 7 PERFORMANCE ANALYSIS AND EVALUATION OF IBMANAGER

7.1 Analysis of Communication Latency

In general, latency is the period of time that one component in a system is spinning its wheels waiting for another component. Latency, therefore, is wasted time. For example, in accessing data on a disk, latency is defined as the time it takes to position the proper sector under the read/write head [86]. In this study, the communications inside the IBmanager are based on event-driven model, in which the events will be fired immediately when the terms are met. These will not introduce latency. In the peripheral communications, there are two communications possible to lead to latency. One is in the communication between the IBmanager and the driver part, the second is in the communication between the IBmanager and its clients.

7.1.1 Latency in Driver Level

In the driver level, latency can be introduced in the polling interval of the remote XML Converter to sub-systems/devices, and the communication between Driver Object and the remote XML Converter as Figure 7.1. The total latency in driver level is the sum of these two ones.

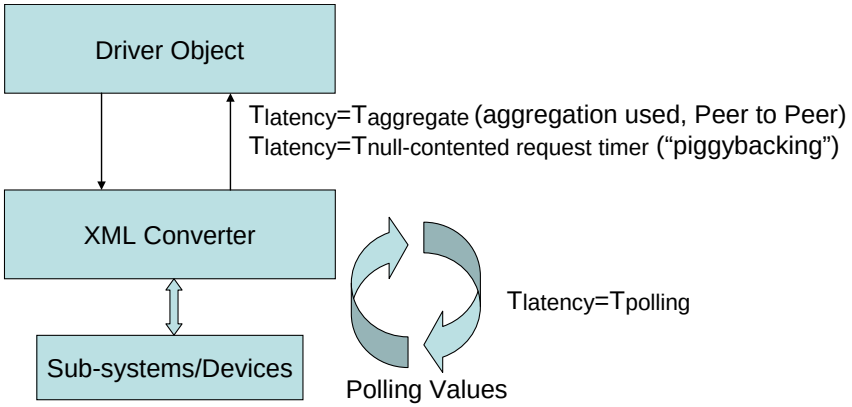


Figure 7.1 Latency in the driver level

The remote XML Converter reads the data points’ values to memory cache in advance by timer-trigger. The lead latency is the interval of polling timer.

T latency=T polling

In the communication between Driver Object and the remote XML Converter, there exist various communication models.

The first one is Peer to Peer method. It is suitable when the two sides both can initiate communication. When Peer to Peer method is used, the latency is the time for aggregating points if the aggregation method is used.

T latency=T aggregate

The second one is “piggybacking” method. When the “piggybacking” method is used, the timer which triggers the null-contented request will result in latency. The interval of the timer is the maxim latency of this method. The “piggybacking” methods is similar to polling method, it has the same latency characteristic of polling

method.

7.1.2 Latency between IBmanager and Its Client

In these communications of IBmanager and its client, “piggybacking” and Peer to Peer methods are used. When Peer to Peer method is used, the latency is the time for aggregating points if the aggregation method is used.

$$T\ latency = T\ aggregate$$

When the “piggybacking” method is used, the timer which triggers the null-contented request will result in latency. The interval of the timer is the main latency of value update of data point.

$$T\ latency = T\ timer$$

7.1.3 Latency Caused by Web Update

As discussed before, there are many technologies to be used in update information in Web page without loading the entire web page. No matter that what technologies are used, the client application (HMI) always needs to request information from the IBmanager. The update of information in Web pages is triggered by timer automatically. The timer-triggered process will result in latency.

7.2 Integration/Interoperation Test of IBmanager Platform

On the basis of the middleware framework, a test integration instance was

deployed in the PolyU IB Lab, which integrated four BA sub-systems from four different vendors in the IB Laboratory. Integration and interoperation of different vendors' systems are realized over campus network as Figure 7.2. Tests were conducted to verify and confirm the integration/interoperability of the middleware framework as well as the technologies associated.

In the IB lab, all the products are connected to the IBmanager. Various interfaces are used, for example, OPC interface by Honeywell EBI and Johnson Controls' M5, DDE interface from Echelon LonMaker, ActiveX interface from (Honeywell) Alerton's Envision. All these interfaces are converted into XML Drivers of the IBmanager. The integration/interoperability of the IBmanager has been tested and worked well.

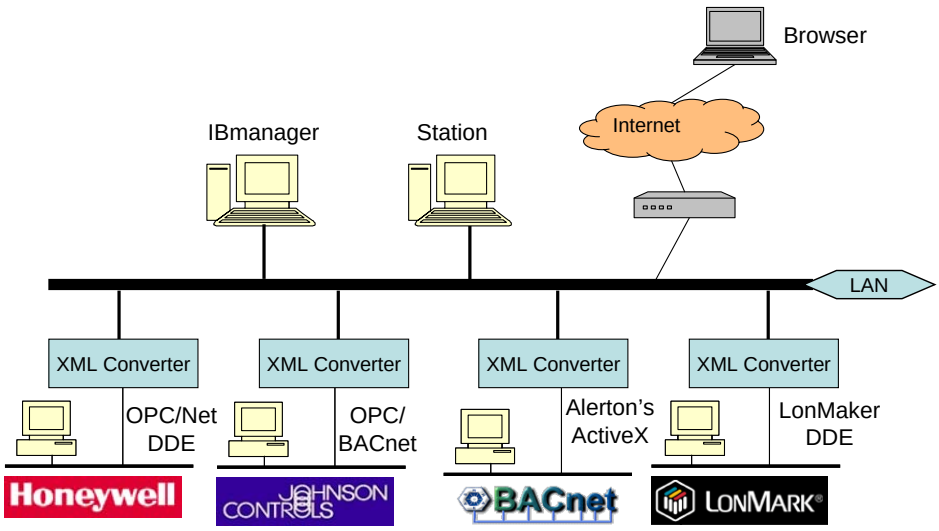


Figure 7.2 Sub-systems integration using IBmanager in PolyU IB Lab

7.3 Roundtrip Time Test of IBmanager Platform

In particular, to measure the response speed of the middleware framework developed, experiments to test the performance of this IBmanager model were conducted as Figure 7.3. In these response speed experiments, we used a “roundtrip time testing client” to read an item in the OPC Server Simulator which connects to the IBmanager. The OPC Server Simulator and the IBmanager are located in the PolyU IB Lab in Hong Kong. The OPC Server Simulator used is a free software package available on Internet [87], which comprises a serial of groups and items. In order to test the roundtrip time in different cases, the tests were conducted in three test environments. **In the first test environment**, the “roundtrip time testing client” was located in Mainland China (Shenzhen) connecting to the Internet via cable modem provided by public services provider. **In the second test environment**, “roundtrip time testing client” was located in a building out of the university campus connected to the Internet via public broadband service. **In the third test environment**, “roundtrip time testing client” was located in a building in the university campus network.

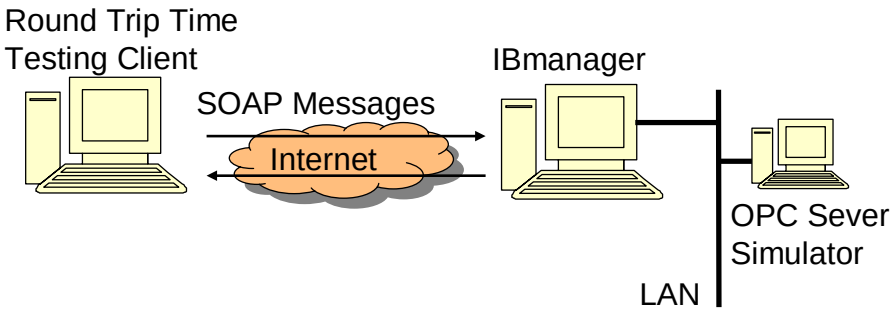


Figure 7.3 Illustration of roundtrip time measurement experiment setup

The values of the roundtrip time measurements are presented in Table-5. Examining the data listed in the table, the roundtrip time of access from Mainland China was 6.85 seconds and 0.71 second, respectively, for the first and successive communications. The roundtrip time of access between the middleware framework and the remote client in Hong Kong was 1.612 second and 0.301 second, respectively, for the first and successive communications. The roundtrip time of access between the middleware framework and the client within the university campus network was 0.738 second and 0.218 second, respectively, for the first and successive communications. It can be observed that the first communication took more time and the successive roundtrip time of Web Service was significantly less. Response speed of the remote access within Hong Kong was noticeably slower than that of the access within the campus network, but the difference is not significant. The response speed of the remote access between Mainland China and Hong Kong was significantly slower. In fact, it is generally believed that efficiency of the Internet connection between Mainland China and Hong Kong is rather low. The efficiency of the Internet connection between USA/Europe and Hong Kong could be much better. Nevertheless, the test results show that the response speed of the middleware framework developed based on Web Services is satisfactory for BAS applications for both remote applications within a city and between different countries.

These results can also be compared with the roundtrip time of the Internet access from client in USA to server in Europe using XML-DA Gateway to OPC DA Server, which belongs to OPC XML-DA – new development of OPC technology. The

reported roundtrip time was 1.2 seconds [88]. Therefore, the response speed of the middleware framework developed in this study is noticeably faster than that from available benchmark besides its benefits in BAS applications.

Table-5 Comparison of roundtrip time of the IBmanager over the Internet

	Average of first accesses (ms)	Average of successive accesses (ms)
Between Mainland China and Hong Kong	6122	673.3
Within Hong Kong	1612	300.5
Within campus network	737.5	218.3
XML-DA to OPC DA Server (between Europe and US) [88]	1200	

7.4 Load Test of the IBmanager Platform

The system load after running the IBmanager has been tested to make sure the system work robust. A typical measurement with 1000 data points is presented as Figure 7.4. In this measurement, data source is one thousand OPC items which is simulated by software. After adding these 1000points to the IBmanager with OPC communication rate 5s, from the CPU usage chart, we can find that the CPU usage and memory usage is reasonable, and the CPU usage fluctuates with 5s interval identical to OPC communication interval.

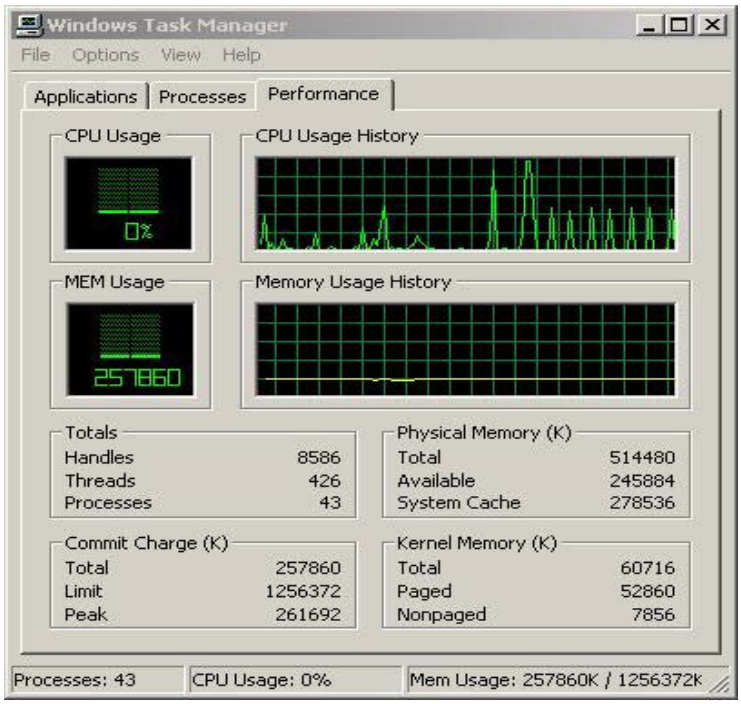


Figure 7.4 System load after running IBmanager

CHAPTER 8 PRACTICAL USE OF IBMANAGER PLATFORM IN A LARGE BUILDING

8.1 Introduction of ICC Project

The IBmanager platform is used in the on-going project “energy efficiency through intelligent control and diagnosis”. This project is one of four application research projects which are funded by the largest local property developer for The Hong Kong Polytechnic University over 5 years. Figure 8.1 is the schematic profile of the building “International Commerce Center” (ICC). This building is super high-rising of 490 meter high above the ground with about 440,000 m2, involving a basement of four floors, a block building of 6 floors and a tower building of 112 floors. The basement is mainly used for car parking with about 24,000 m^2. The block building from the ground floor to 5th floor mainly serves as commercial center involving restaurants, shopping markets and exhibition halls. The gross area is about 67,000 m^2. For the tower building, the 6th and 7th floors serve as mechanical floor to accommodate chillers, cooling towers, pumps etc. The 8th is refugee floor. From 9th to 98th floors, there are mainly commercial office floors with each floor of length 66 m and width 65 m except that the 41st and 77th floors are used as refugee floors, and the 42nd, 78th and 99th floors are used as mechanical floors to accommodate mechanical equipments such as heat exchangers, pumps, PAU and fans etc. A high-graded hotel is located from the 100th to 118th floors.

Figure 8.1 A view of the ICC building

In this project, we are required to implement sophisticate control strategies and propose more energy efficient control strategies to saving more energy while maintaining the indoor environmental requirement on the basis of original design of a being constructed super high rising commercial office building. These strategies cover chiller system, water side systems and air-side systems etc. To apply the control strategies, some measurement instruments are needed to add into the original systems to allow more effective and efficient control. These measurements are integrated to the intended **ATC** (automatic temperature control) system and building management system (BMS). The software of control strategies and supervisory control and the corresponding hardware are also needed to be connected or be parallel to the originally intended control platforms [89].

As the support platform of the control strategies and supervisory control in the ICC project, IBmanager is customized for the application. The platform reads the status data of HVAC system, invokes the optimization software to calculate the optimized parameters, and then transfers the optimized parameters to the originally intended control platform. The entire optimization task is done by online mode. The change of status data may lead to the change of running parameters, hence to make the overall HVAC system works in an energy-saving status. This process is conducted online by an automatic mode.

8.2 Current Status and Difficulties of Optimization Application

As reviewed before, in the current real application project, the HVAC optimization application is mainly restricted in offline application. It calculates and makes some suggestions for facility managers based on the historical data. However, the environment (for example, weather) may be changed fast, the offline optimization software cannot suggest and adjust the parameters timely.

With the support of the IBmanager platform and research work of other colleagues in HVAC optimization group of BSE Department in The Hong Kong Polytechnic University, IBmanager realizes the online optimization application of HAVC system. These colleagues have developed practical simplified optimization calculation, which is implemented on the IBmanager platform with the support of the flexible interfaces and unified data format.

8.3 Architecture of the BA System Used in ICC

Figure 8.2 shows the implementation architecture of ICC optimization project, which includes control optimizers and the robust control strategies of air systems and chiller plant. All the DDC controllers of air systems (including PAU, AHU and VAV terminals) are integrated into a LAN-based BMS.

Supply air control optimizer is to optimize the temperature set point and static pressure set point with minimum energy consumption while keeping comfortable temperature and humidity environment as well as enough air circulation. The optimizer will be programmed in AHU local control stations because of not very complicated programming and demand on computation power. ***Fresh air control optimizer*** is to optimize the fresh air flow rate of each AHU. Optimal fresh air intake can guarantee acceptable **IAQ** (Indoor Air Quality) with minimum energy consumption. The control optimizer will also be programmed in same local control station. The control/optimization logic and formulas will be provided by the group of PolyU. The HVAC&BMS contractors implement the strategy in the programmable control stations with the support of the group of PolyU [89].

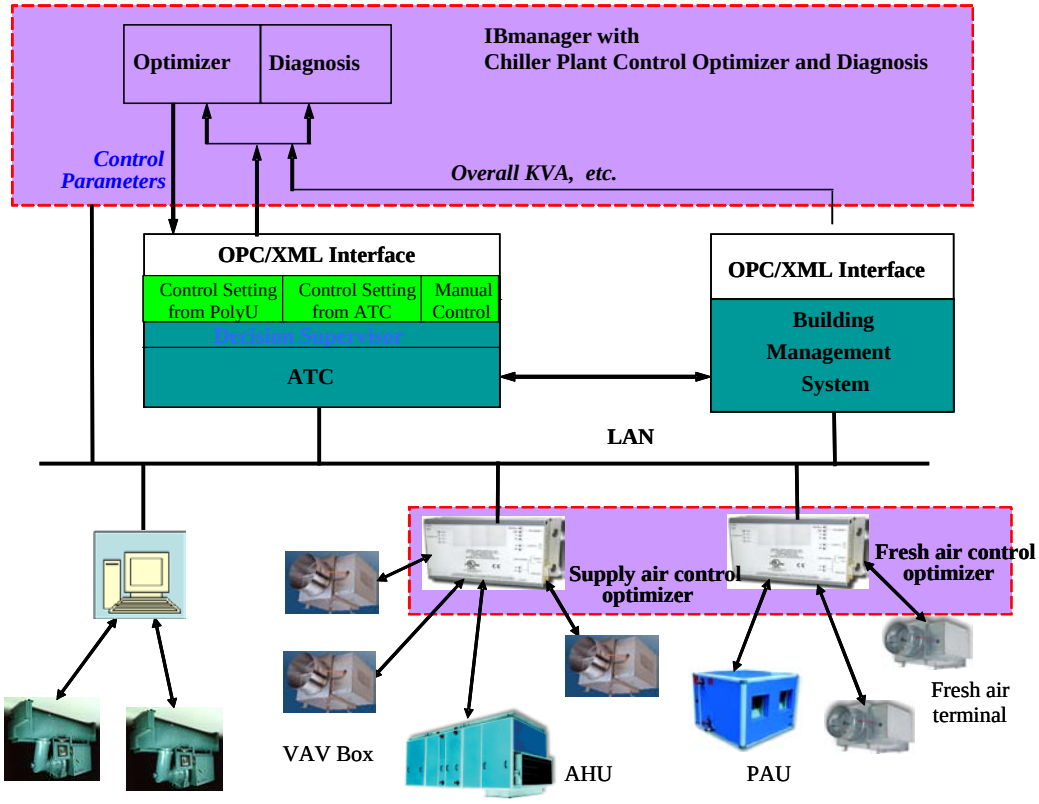


Figure 8.2 Implementation architecture of ICC project

The ***robust chiller sequencing control strategy and chiller plant optimizer*** include the supervisory control of chillers and pump as well as cooling towers. It is done by a complicated program with high demand on computation power. Therefore, a standalone control and optimization software package and a diagnosis package are developed running on a PC station interfaced with the main station of the chiller control system (BMS), as the control/optimization and diagnosis need a great deal of the plant operation information. The standalone package will run in parallel with the chiller sequencing program provided by the HVAC&BMS contractors. The contractors provide the protocol or an interface for the communication between these packages and the main station of chiller control system. The control parameters of the

optimizer mainly involves numbers of chiller, cooling tower, and pump to be operated, the set-point of supply chilled water temperature, the set-point of supply cooling water temperature, the set-point of water pressure differential of the worst water loop etc. When plant (chiller, cooling tower, pumps, etc.) sequencing is of concern, the chiller plant optimizer provided by PolyU will only provide the number of them to be operated and the chiller control system (ATC) will determine which one is used. A decision supervisor in the chiller control system is designed for the operators to set if the settings given by the “chiller plant control optimizer” are used or ignored (not used). The ***chiller performance monitoring and diagnosis strategy*** will be implemented in a standalone package probably running in the same PC station as the optimization package. This package will provide diagnosis information to management staff and will not feedback to chiller control. Both the optimizers and diagnosis packages are realized based on the middleware platform designed in the thesis and research – the IBmanager [89].

8.4 Simulation and Test in Lab

In order to validate the optimization software based on the IBmanager, the optimization project is conducted as two stages as Figure 8.3. One stage is the simulation stage in the PolyU IB Lab environment, the second stage is the deployment in the ICC project site. In the simulation stage, the **IBmanager** communicates with the Virtual Building software which is developed by Prof. Wang based on simulation software - TRNSYS. The IBmanager passes data to and gets response from the Virtual

Building software just like it is interacting with a real building system. In the ICC site commissioning stage, the IBmanager communicates with and acquires data from the original ATC and BMS. In either environment, all the data interested are transformed into common Data Point Objects. The optimization strategy is calculated and realized with Matlab [95]. The Matlab code is compiled as Dynamic Link Library (DLL) to be invoked by the IBmanager. The IBmanager passes data to the Matlab DLL, after the calculation, the result returns to the IBmanager. The return value is mapped as "virtual" common Data Point Objects. These virtual Data Point Objects can be accessed like actual Data Point Objects, for displays or calculations although they are not corresponding to actual data points in the physical systems. As the optimized set-points, these calculated values of the virtual Data Point Objects will influence and optimize the running of the physical systems.

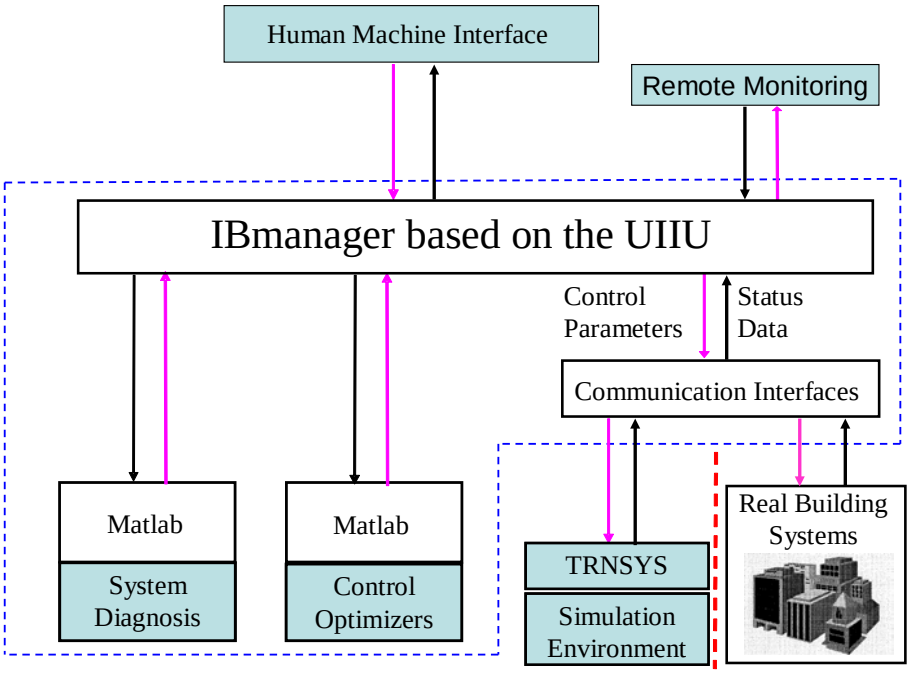


Figure 8.3 Two research environments in ICC project for IBmanager

In the simulation environment, Virtual Building software is used to simulate the characteristics of a building, including inside temperature, inside humidity. These data can be read by communication interface, just similar to read data from a real BMS of a building as Figure 8.4. In order to be similar to a real building automation system, the Virtual Building software is kept online to IBmanager that means, IBmanager reads data frequently from the Virtual Building software, and send new parameters to the Virtual Building software after the calculation gives out consequence. The Virtual Building software will instantly simulate new environment of the virtual building according to the new parameters. Data of the new environment simulated is sent to IBmanager instantly. This is a real-time online system. The sole difference to actual BMS is the communication interface. In the simulation, the interface to transfer data between IBmanager and the Virtual Building software is text file.

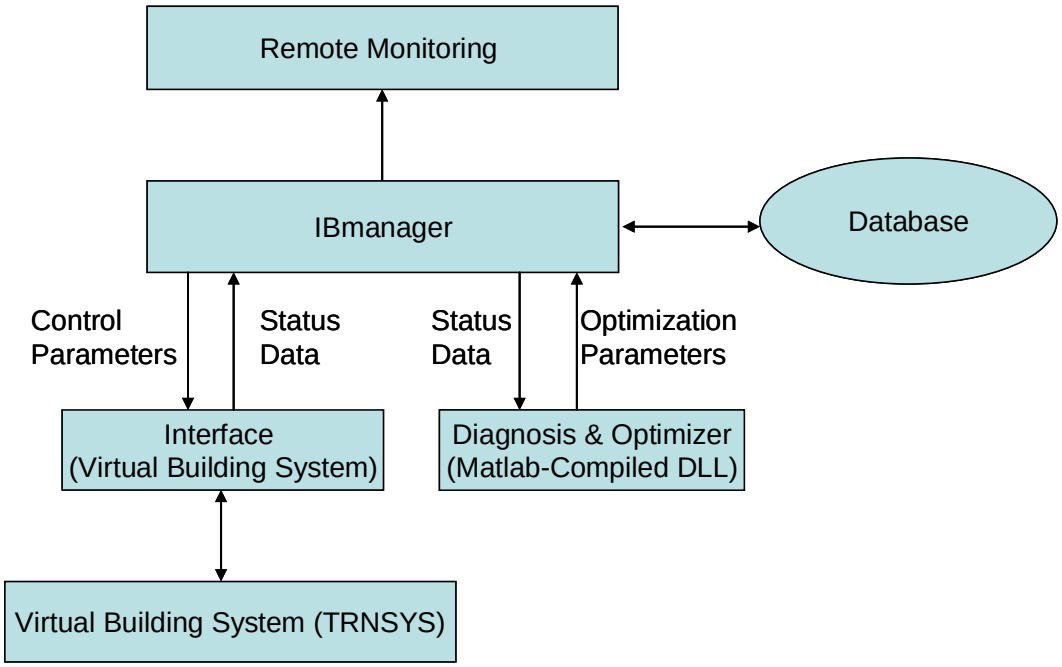


Figure 8.4 Simulation environment of IBmanager in ICC project

8.5 Functions Realized in ICC Project

The IBmanager platform has been provided with a few drivers as Figure 8.5. The IBmanager can communicate with sub-system/devices by these drivers. New sub-systems can be added into the IBmanager if the corresponding driver is developed compliant with specification. After the corresponding driver is developed, the sub-system can be easily added into the IBmanager.

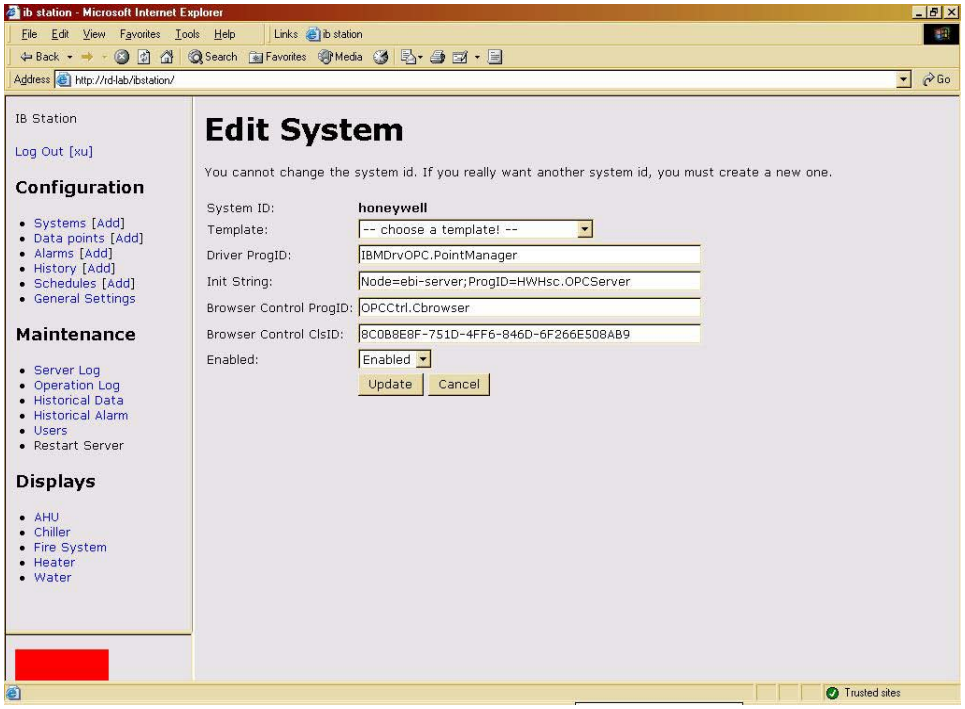


Figure 8.5 Adding sub-systems

After the sub-system is added into the IBmanager, users can browse and add/edit/remove data points within the driver as Figure 8.6. Users can give a human-friendly alias to the raw data point name for convenient management.

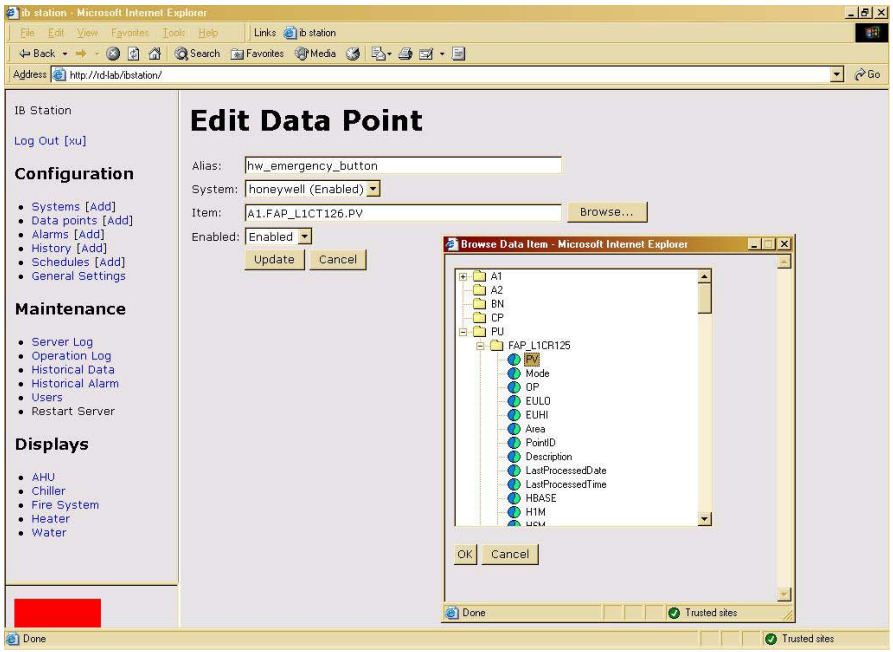


Figure 8.6 Browsing and adding data points

When the data points from different sub-systems are added into the IBmanager, all these data points have the same elements/structure. Regardless which sub-system a data point is from, it has been capsulated as the same common Data Point Object like all other data points. Its human-friendly alias is its identification instead of its original raw name. All the Data Point Objects can see each other and interoperate. Users can make a single graphics display which monitors the data points from various sub-systems as Figure 8.7.

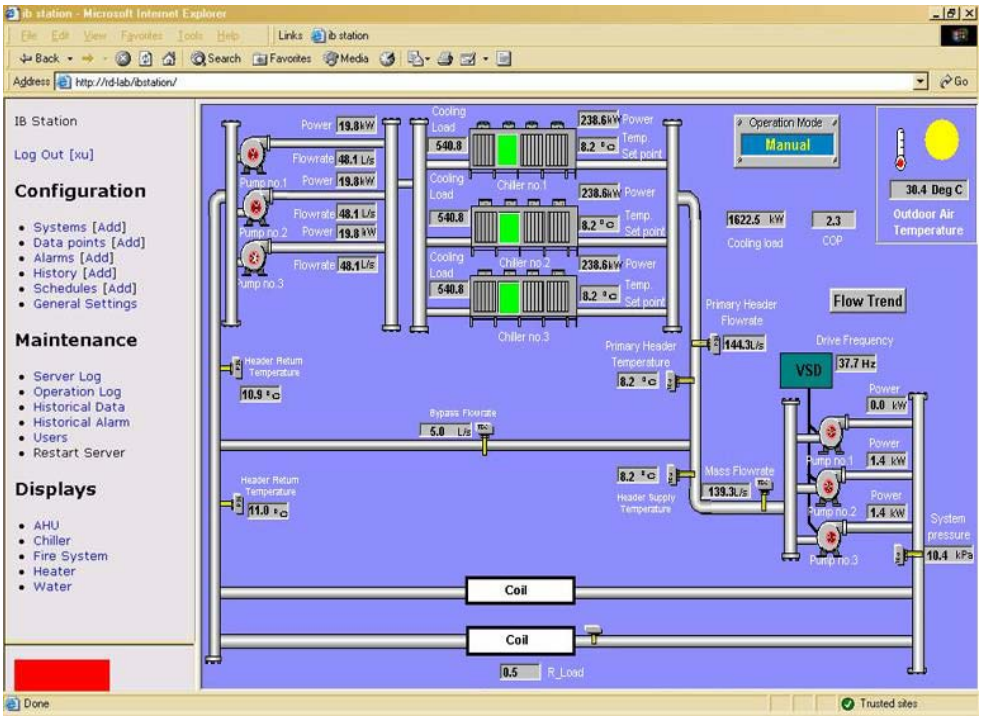


Figure 8.7 Accessing data points of various sub-systems within one display

Users can define the interoperation between the common Data Point Objects from various sub-systems as Figure 8.8.

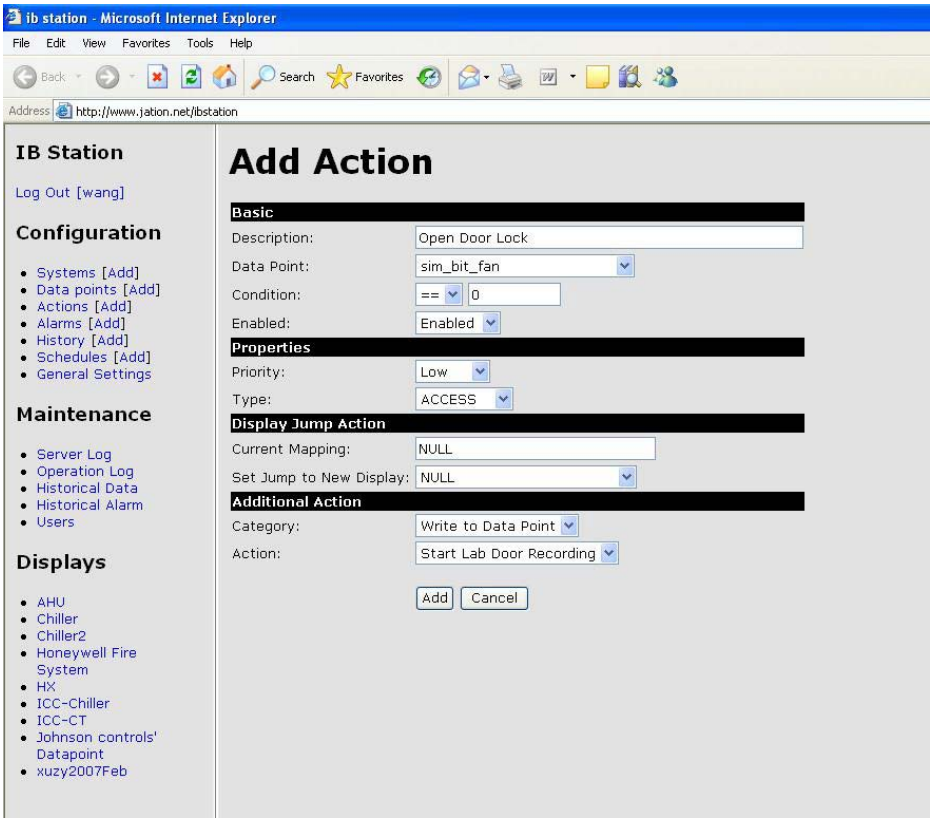


Figure 8.8 Interoperation between various sub-systems

The historical data of data points from various sub-systems has been saved as the same structure in database as Figure 8.9.

h_log_id	h_id	h_value	h_date
234		28	2007-3-2 16:30:26
235		27	2007-3-2 16:31:06
236		26	2007-3-2 16:31:46
237		25	2007-3-2 16:32:26
238		24	2007-3-2 16:33:06
239		23	2007-3-2 16:33:46
240		22	2007-3-2 16:34:26
241		21	2007-3-2 16:35:06
242		20	2007-3-2 16:35:46
243		19	2007-3-2 16:36:26
244		18	2007-3-2 16:37:06
245		17	2007-3-2 16:37:46
246		16	2007-3-2 16:38:26

Figure 8.9 Common historical data structure of data points of various sub-systems

The alarm will be triggered when the trigger term is met. The alarm conditions can be defined and attached to any data point as Figure 8.10.

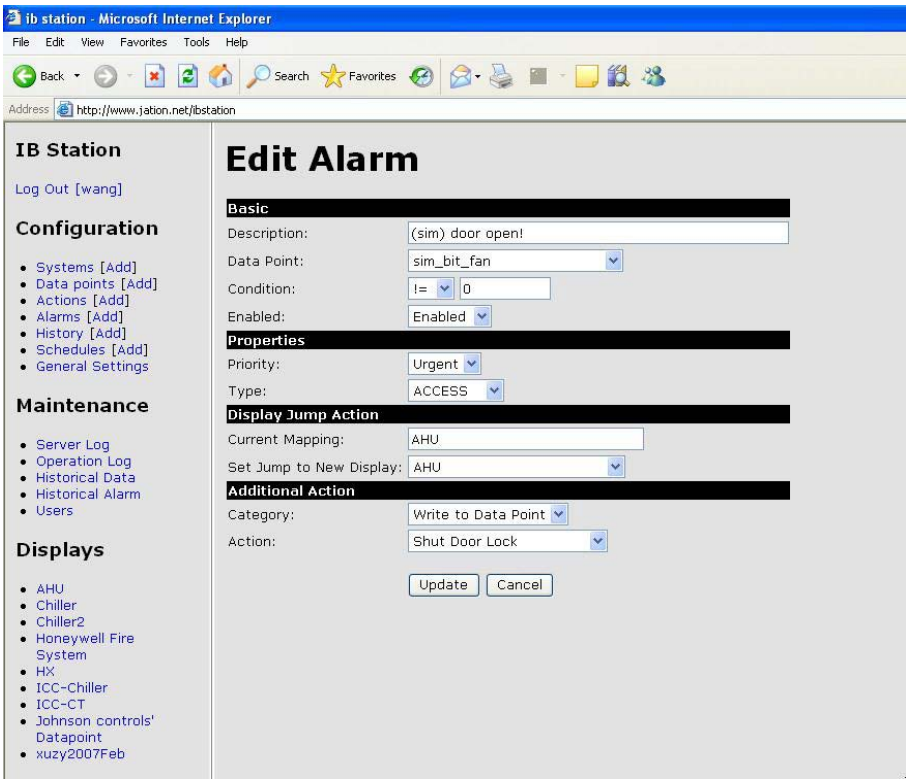


Figure 8.10 Definition of alarm

CHAPTER 9 CONCLUSIONS AND DISCUSSIONS

9.1 Conclusions

The integration platform presented in this thesis employs standard communication protocol and distributed computing technologies, including object-oriented programming, data-subscription & event-driven technology, Web Services, XML driver technology, and value-added services plug-ins technology，to realize data and services integration and interoperation among distributed BASs on the Intranet/Internet.

Support platform for integration applications

The IBmanager can accommodate the heterogeneous sub-systems compliant with diverse communication protocols, provide a unified data model and interface for integration/interoperation applications and value-added services. By this platform, the BA field devices are not only integrated to provide real-time and historical data information to people and the enterprise, but interoperate each other. With the standard open technologies, this platform will communicate with enterprise system and be thought as part of the enterprise system.

Accommodate various communication interfaces

With the XML driver technology, IBmanager can accommodate various communication interfaces, including open protocols and proprietary protocols. With

the standard Web Service technology, the communication scope of the drivers has been extended to the Internet. Users can develop their own drivers for the IBmanager, according to the driver specification provided.

Capable to develop customized value-added services

Users can develop their value-added services to be integrated into the IBmanager. These value-added services will be loaded into the IBmanager as plug-ins, running in the same process with the IBmanager. This make the value-added services can be run online with good performance. As mentioned above, a few optimization services, including FDD and data fusion have been implemented based on the IBmanager.

Unification and uniformity for deployment

The IBmanager realizes an autonomous, self-contained unified integration unit (UIU). It can be deployed in flexible ways, standalone application, vertical chain-type deployment and horizontal chain-type deployment. In different deployment architecture, all IBmanager installations have the identical characteristics, this decreases the training and maintenance cost of multiple installations at large-scope applications.

Extensibility and scalability

The IBmanager is designed consisting of batch of function objects that keep the systems to have good extensibility. These objects can be instantiated and loaded as per

the requirement of projects. The Data Point Objects are the core components of the overall platform, which drive the high-level functions running. Data-subscription & event-driven methods are mainly used for the communication between function objects. Only the events subscribed by other objects/components will be fired when the term is met, and only the subscribers will handle and respond the events they subscribe. This makes the system can accommodate large amount of data points without scarifying performance significantly.

Unified standard communication technology

The platform adopts standard Web Services as its communication technology. The bi-directional communication method is designed based on Web Service, this makes the transportation of COV and A/E messages based on Web Service feasible. The Web Service technology is used for the communication between the platform and the driver, the communication between the platform and its client, and access to the remote database. Hence Web Service communication becomes the unified communication interface among the platforms, remote database, drivers and their clients. The platform can integrate other Web Services, meanwhile provides Web Services interfaces which can be invoked by other IBmanager installations, or other customized client applications.

Various sub-systems/devices can be integrated into the IBmanager with the designed XML Driver technology. Based on this technology, different protocols can be converted to common XML-encoded messages. The sub-systems/devices can be

anywhere on the Internet since the XML-encoded message can be transported by Internet standard protocol - HTTP. This expands the driver communication beyond the LAN.

Heterogeneous databases integration

The remote heterogeneous databases can be integrated by Web Service. Every remote database is wrapped as a Web Services provider. Based on the design of Database Agent Object, a unified database access interface for the remote heterogeneous databases is designed.

With the help of the powerful communication capability of Web Services on heterogeneous platform and on the Internet, the middleware technology becomes a seamless integration platform on the Internet. By the XML Driver and the Web Services interfaces presented by the IBmanager, the designed IBmanager installations can be deployed as chain-type architecture. The IBmanager can work standalone as a full-function BMS, or just as a part of large-scale BMS applications.

Services integration interface with enterprise applications

The middleware platform might be used to integrate other services, for example, weather report service. HVAC systems can optimize their control according to the data obtained from Web Services of the (government) Weather Bureau. Maintenance application can read specification data and manual of devices from manufacturers. Meanwhile the IBmanager middleware platform might be integrated by other

applications on the Internet. For example, FDD applications can read real time data and historical data from the IBmanager by Web Services interfaces. By the same way, ERP (Enterprise Resource Planning) application can obtain the energy consumption of the entire enterprise from Web Services interfaces of the IBmanager. It ensures complete integration and interoperability among diverse facility systems and devices by connecting them to each other, to enterprise systems, and to the Internet in real time mode. This allows personnel using a standard web browser to measure, manage, and control a wide variety of energy, building, and security applications from anywhere in the world.

System performance and actual applications

The system performance has been discussed and measured primarily. The communication latency resulted from drivers and client communication has been discussed, the roundtrip time of the communication between the IBmanager and its client has been measured.

As an integration platform, users can develop their own applications based on the IBmanager. A practical HVAC optimization application is on-going on a big-scale project - the International Commercial Center (ICC) project. The IBmanager provides integration and value-added services platform support for this project.

9.2 Discussions and Future Work

The integration platform is still on development process. Some works may be

done in the future to enhance it functions.

9.2.1 Load-balancing

It is needed to design and implement the load-balancing scheme for the IBmanager in a real wide-area environment. For example, multiple IBmanager installations can be deployed, load can be moved from one IBmanager installation to another to keep load-balancing among various IBmanager installations, when some IBmanager installations are heavily loaded or not working properly. Measurements of load will be defined to provide an assessment for the load-balancing schemes. Load distribution strategy will be determined according to the measurements. For example, if faster communication service is required by data points, a load measurement that puts higher weight on communication bandwidth will be selected. Thus, an IBmanager installation with greater communication bandwidth can be selected to connect to the data points with more possibility.

9.2.2 Mobile application

The mobile applications include various media and applications, for example, SMS alarming, accessing the BMS by PDA (personal digital assistant) or mobile phone. By PDA/mobile phone applications, user can make client using Java 2 Micro Edition to provide UI (user interface) to monitor the BMS, or browse the web pages by embedded browser. The traditional protocol is WAP, now *html* has got broader support in PDA/mobile phone.

9.2.2 Security of Web Service

The security of Web Service is not implemented in the IBmanager platform yet. At the beginning of the research, the security of Web Services has become a hot topic to users and organizations. Because it is based on program-to-program interactions as opposed to human-to-program interaction, it is important for Web Service security to address topics such as access control, authentication, data integrity and privacy. Today the most common security scheme is SSL (Secure Sockets Layer), but when it comes to Web Services there are limitations with SSL. The Web Service technology has been moving towards different XML-based security schemes for Web Services. Some of the XML-based security technologies include the following [91]:

- XML digital signature
- XML Encryption
- XKMS (XML Key Management Specification)
- SAML (Secure Assertion Markup Language)
- WS-Security (Web Services Security) [90]
- ebXML Message Service [91]

The implementation of security is an important consideration for the IBmanager in the future development.

9.2.3 Public Services

In the future development, more services may be developed based on this

platform, such as data-mining, building energy analysis. Anyone with interest can develop their own value-added services based on the IBmanager. These value-added services can be placed on the Internet for public access.

References

1. Ehrlich P. “Guideline for XML/Web Services for Building Control”, BuilConn 2003, Dallas, Apr. 2003
2. OASIS Open Building Information Exchange (oBIX) TC, oBIX FAQ, http://www.oasis-open.org/committees/obix/faq.php as viewed on June 11, 2008
3. Oswald P., Rockwell G., “Bringing the Power of the Internet to Real-time Control and Automation Systems”, *AutomatedBuildings.com* Article, July 2000;
4. Wang S.W. and Xie J.L. "Integrating Building Management System and Facility Management on Internet", *Automation in Construction*, V11(6), pp. 707-715, 2002
5. Wang S.W., Xu Z.Y., Li H., Hong J. and Shi W.Z. “Investigation on Intelligent Building Standard Communication Protocols and Application of IT Technologies”, *Automation in Construction*, V13(5), pp. 607-619, Sep. 2004
6. Wang S.W, Xu Z. Y., Cao J.N., and Zhang J.P., “A Middleware for Web Service-enabled Integration and Interoperation of Intelligent Building Systems”, *Automation in Construction*, V16(1), pp.112-121,Jan. 2007
7. ANSI/ASHRAE Standard 135-2001: BACnet® - A Data Communication Protocol for Building Automation and Control Networks, Atlanta Georgia: American Society of Heating Refrigerating, and Air-Conditioning Engineers, 2001
8. ASHRAE SSPC 135, Official Website of ASHRAE SSPC 135, *http://www.bacnet.org/* as viewed on 2008-5-17;
9. EIA Standard “Control Network Protocol Specification” EIA/CEA-709.1-B (Revision of EIA-709.1-A), Jan., 2002
10. Fisher D.M., "BACnet & LonWorks: A White Paper", July, 1996, *http://www.bacnet.org/*
11. The Cover Pages Web Site, “ASHRAE Releases BACnet Web Services Interface Specification for Public Review“, http://xml.coverpages.org/ni2004-10-22-a.html, October 22, 2004.
12. Echelon Corporation, “i.LON 100 e3 User's Guide”, http://www.echelon.com/support/documentation/manuals/cis/078-0310-01B.pdf as viewed on June 11, 2008
13. Frost and Sullivan Co., "North American Building Automation Protocol Analysis", *Frost and Sullivan Report A143-19*，May, 2002
14. Huang H.Y., Yen J.Y., Chen S.L., Ou F. C., Development of an intelligent energy management network for building automation, *Automation Science and Engineering, IEEE Transactions*, V1(1), July 2004, pp. 14 - 25
15. Xiao F., Sensor Fault Detection and Diagnosis of Air Handling Units, *thesis for Doctor of Philosophy*, at The Hong Kong Polytechnic University in April, 2004
16. Huang G.S., Wang S.W. and Sun Y.J., Improving reliability of chiller sequencing control using fused measurement of building cooling load, *HVAC&R Research*, In

Press

17. Craton E., Robin D., “Information Model: The Key to Integration”, *AutomatedBuildings.com* Article, January 2002, http://www.automatedbuildings.com/news/jan02/art/alc/alc.htm as view June 11, 2008;
18. Fisher D. M., “XML, Web Services, and the Problems of Enterprise-Level Data Exchange,” HPAC Engineering, vol. 76, no. 4, pp. 13–14, April 2004
19. Overview, OASIS Open Building Information Exchange (oBIX) TC, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=obix as viewed on May 12, 2008;
20. ANSI/ASHRAE Addendum c to ANSI/ASHRAE Standard 135-2004, American Society of Heating, Refrigerating and Air-Conditioning Engineers, Atlanta, 2006
21. What is M2M Expo & Conference? http://www.m2mexpo.com/content/whatism2m.asp?tab=a as viewed on June 11, 2008;
22. Bushby S.T., "Communication Gateways: Friend or Foe? ", *ASHRAE Journal* V40(4), pp. 50-53, April, 1998
23. Zheng F.W., Pu H.M., Huang D.M., Integrating Intelligent Building with CORBA, *Microelectronics and Computer*, V21(5), pp.25-29, May 2004
24. Sinclair K., “Corporate Enterprises Are Forever Changed With Real-Time Building Information”, May 2005, http://www.automatedbuildings.com/news/may05/articles/esarticle/sinclair.htm as viewed on June 11, 2008;
25. McGowan J., “Expanding Horizons for System Integration”, *AutomatedBuildings.com* Article, January 2002, http://www.automatedbuildings.com/news/jan02/art/mcg/mcg.htm as view June 11, 2008;
26. Ehrlich P., “Buildings - Part of the Enterprise”, Enterprise@BuilConn, March 23, 2005, http://www.builconn.com/agenda/track_overview.asp?qsTID=29&qsEID=9&qsstyle=1 as viewed on June 11, 2008
27. Ehrlich P. and Sinclair K., EMAIL INTERVIEW, Enterprise@BuilConn, AutomatedBuildings.com Interview, February 2005, http://www.automatedbuildings.com/news/feb05/interviews/ehrlich.htm as view June 11, 2008;
28. Tom S., Web services and BACnet, *ASHRAE Journal*, v 46, n 10, October, 2004, p S14-S17
29. Bakken D. E., MIDDLEWARE, Washington State University, *Encyclopedia of Distributed Computing, Kluwer Academic Press*, 2003;
30. Microsoft Corporation, “Understanding the Distributed Object Component Model (DCOM) Architecture”, *http://www.microsoft.com/ntserver/techresources/appserv/COM/DCOM/1_Introdu*

ction.asp as viewed on October 10, 2004

31. Object Management Group, *http://www.omg.org/* as viewed on June 11, 2008
32. Java Remote Method Invocation - Contents, *http://java.sun.com/j2se/1.4.2/docs/guide/rmi/spec/rmiTOC.html* as viewed on June 11, 2008
33. Raj G. S., "A Detailed Comparison of CORBA, DCOM and Java/RMI", http://my.execpc.com/~gopalan/misc/compare.html as viewed on May 14, 2008;
34. MIMOSA, "Open Systems Architecture for Condition-Based Maintenance web-based training, Middleware Technologies", http://www.osacbm.org/Documents/Training/TrainingMaterial/TrainingWebsite/c1p4.html as viewed on June 11, 2008
35. Gisolfi D., Web Services architect, Part 3: Is Web Services the reincarnation of CORBA? 01 Jul 2001 http://www.ibm.com/developerworks/webservices/library/ws-arc3/ as viewed on June 11, 2008
36. Box D., A Young Person's Guide to The Simple Object Access Protocol: SOAP Increases Interoperability Across Platforms and Languages, http://www.neovis.pe.kr/AspNet/Lec/download.aspx?file=SOAP.pdf as viewed on June 11, 2008
37. Würth D., "Dynamic Service Discovery across Technology Boundaries, A Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science", http://www.emn.fr/x-info/emoose/alumni/thesis/dwurth.pdf as viewed on June 11, 2008
38. School of Computer Science and Engineering , BeiHang University, Chapter 4: Message-Driven Bean, Feb. 2006 http://act.buaa.edu.cn/Download/ch4%20Message-Driven%20Bean.ppt as viewed on June 11, 2008
39. Maechling P., UNAVCO/IRIS Web Services Joint Workshop, 8 June 2005 http://www.iris.edu/workshops/2005stevenson/ws_presentations/Maechling_WS_Workshop.ppt as viewed on June 11, 2008
40. Lu N., Liang J., You J. Y., Dept. of Computer Sci. and Eng. Shanghai Jiaotong Univ., "Design and Implementation of Building Intelligent Management System", *Journal of Shanghai Jiaotong University*, July, 2000
41. Guo H., Chen R., Hu L.M., "Research on integrated model based on XML, CORBA and agent", *Proceedings of the International Conference on Computer Supported Cooperative Work in Design*, pp.344-348, July, 2001
42. Wang Y., Wang Z.Y., "Design and Implementation of Intelligent Building Management System (IBMS) based on CORBA", *Computer and Digital Engineering*, vol29(2), pp.16-22, 2001
43. Johnson Controls Inc., "Product Information", *http://cgproducts.johnsoncontrols.com/tree.asp* as viewed on October 10, 2004

44. The World Wide Web Consortium, http://www.w3.org/ as viewed on October 10, 2007
45. Schmidt D. C., “Object Interconnections: CORBA and XML - Part 3: SOAP and Web Services”, 2001, http://www.ddj.com/cpp/184403802
46. Carnegie Mellon Software Engineering Institute, “Object Request Broker, Software Technology Roadmap”, http://www.sei.cmu.edu/str/descriptions/orb.html
47. Vasudevan V., “A Web Services Primer”, April 04, 2001, http://webservices.xml.com/pub/a/ws/2001/04/04/webservices/
48. Donlon M., “Standard Internet Protocols in Building Automation”, *Engineered Systems*, V19(2), pp61-68, Feb., 2002
49. The Continental Automated Buildings Association (CABA), “CABA XML/Web Services guideline committee states mission & objectives” July 15, 2003, *http://www.caba.org*
50. OASIS Open Building Information eXchange Technical Committee, “History of oBix”, http://www.obix.org/ as viewed on October 10, 2004
51. Newman H. M., "BACnet/XML: BACnet Opens Doors for New Technology." *ASHRAE Press Briefing for BACnet*. January 27, 2004, revised January 28, 2004. http://www.ashrae.org/publications/detail/14844
52. McGowan J., “DDC Networks: …is the question really BACnet or LonWorks?”, *AutomatedBuildings.com* Article, Sept 2000, http://www.automatedbuildings.com/news/sep00/articles/jmcg/jmcg.htm as viewed on 2008-5-17
53. Sidebar on Web Services, http://www.automatedbuildings.com/news/sep02/articles/stom/stom.htm as viewed on 2008-5-17
54. Wikimedia Foundation, Inc., “Wikipedia, Heterogeneous Database System”, http://en.wikipedia.org/wiki/Heterogeneous_Database_System#Technical_Heterogeneity
55. Kranz H. R. & Gisler O., “How standardization and IT technology will shape the BACS industry in the future”, *AutomatedBuildings.com* Article, January 2002, http://www.automatedbuildings.com/news/jan02/art/hk/hk.htm
56. FieldServer Technologies, *http://www.ahrexpo.com/showpreview/buildingautomationcontrol.php*, February 7-9, 2005
57. OPC Unified Architecture, http://www.opcconnect.com/ua.php as viewed on 2008-5-17
58. Claus E., "Synchronic Extending interoperability and connectivity", *AutomatedBuildings.com* Article, March, 2002, http://www.automatedbuildings.com/news/mar02/art/isys/isys.htm as viewed on June 11, 2008

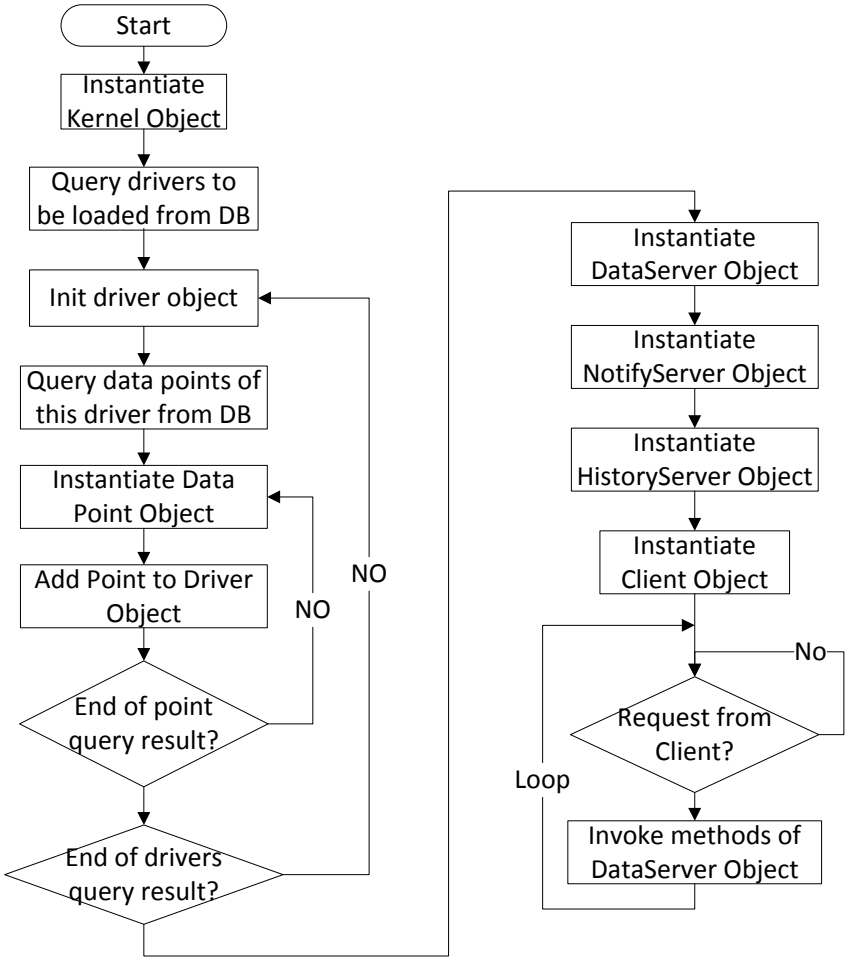
59. Luth, J., "OPC Brings XML-DA to the Factory Floor," *Control Solutions*, Vol. 75(7), pp. 34-36, 2002.
60. Baumann G. and Damm M., "Industrial Communication System-independent and flexible", PRAXIS Profiline - OPC, Vogel Verlag, Wuerzburg, No. 1, pp.43-46, *http://www.is.siemens. de/data/presse/docs/isfb01033112e.pdf*, Jan., 2003
61. OPC Foundation, "OPC Foundation's Unified Architecture to be Released at ARC Forum", Archive Press Release, June 27, 2006, http://news.thomasnet.com/companystory/489072
62. Desmarais R., "XML Esperanto for Data", *AutomatedBuildings.com* Article, May 2000, http://www.automatedbuildings.com/news/may00/articles/teletrol/ttrol.htm as viewed on June 11, 2008
63. Wolter R., "Extreme XML Simply SOAP", October 15, 2001, http://msdn.microsoft.com/en-us/library/ms950803.aspx as viewed on June 11, 2008
64. Wikimedia Foundation Inc., "SOAP, From Wikipedia", http://en.wikipedia.org/wiki/SOAP as viewed on June 11, 2008
65. Microsoft Corporation, "What's New in Enterprise UDDI Services", July 17, 2003, http://www.microsoft.com/windowsserver2003/evaluation/overview/dotnet/uddi.mspx as viewed on June 11, 2008
66. OASIS Forms Open Building Information Exchange (oBIX) Technical Committee, News: Cover Stories, May 11, 2004, http://xml.coverpages.org/ni2004-05-11-a.html as viewed on June 11, 2008
67. OASIS Open Building Information Exchange TC, "oBIX 1.0, Committee Specification 01", OASIS, 5 Dec. 2006
68. Tom S., "Web Services A New BACnet Standard", December 2004, *AutomatedBuildings.com* Article, http://www.automatedbuildings.com/news/dec04/articles/alc/stom.htm as viewed on June 11, 2008
69. Hypertext Transfer Protocol -- HTTP/1.1, HTTP Message, http://www.w3.org/Protocols/rfc2616/rfc2616-sec4.html as viewed on June 11, 2008
70. Microsoft Cooperate, SOAP Request Message Structure, http://msdn.microsoft.com/en-us/library/ms190796.aspx as viewed on June 11,2008
71. Iwanitz F., "Technical Article: XML-DA opens windows beyond the firewall", http://ethernet.industrial-networking.com/articles/articledisplay.asp?id=21 as viewed on June 11, 2008
72. W3C UK and Ireland Regional Office, "Demonstration: Query Multiple Databases using Semantic Web Technology", http://www.w3c.rl.ac.uk/QH/WP1/demo.html as viewed on June 11, 2008
73. Carroll N. L., Calvo R. A., "Querying Data from Distributed Heterogeneous Database Systems through Web Services",

http://ausweb.scu.edu.au/aw04/papers/refereed/carroll/paper.html as viewed on June 11, 2008

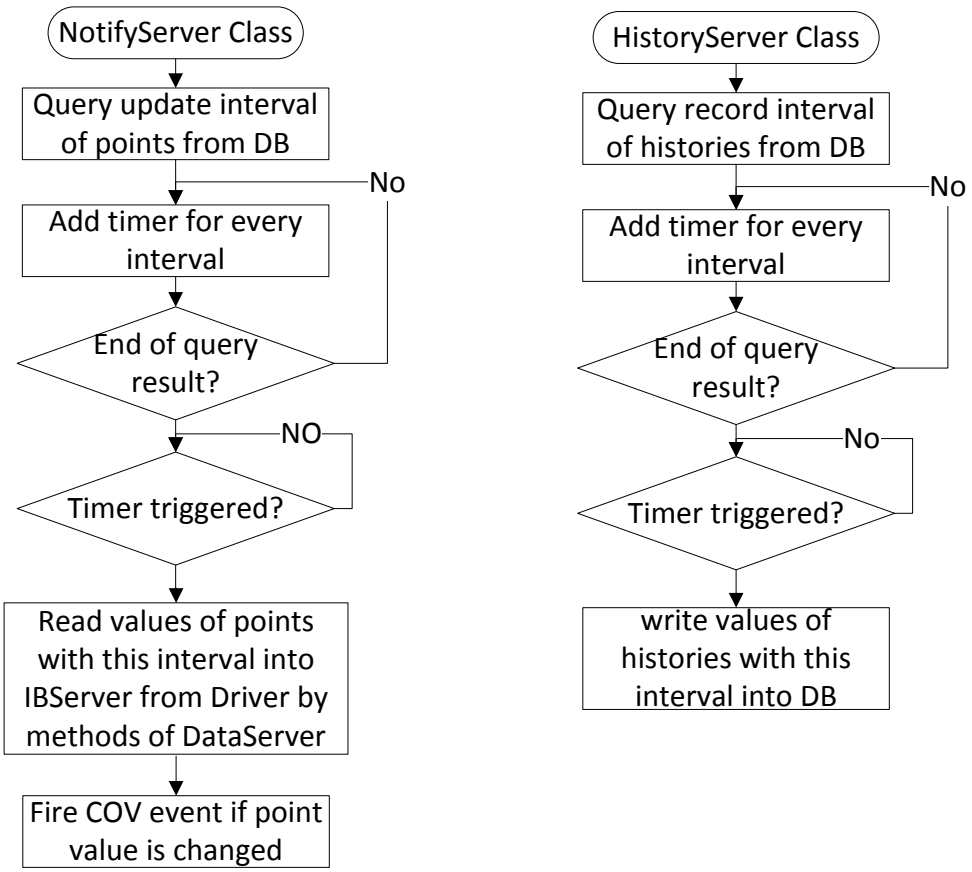
74. Nikitas M., “Improve Web Services' Performance by Compressing SOAP”, Jan. 2003, http://dotnetjunkies.com/Article/46630AE2-1C79-4D5F-827E-6C2857FF1D23.dcik as viewed on June 11, 2008
75. Joshi B., “Accessing XML Data using ASP”, http://www.4guysfromrolla.com/webtech/101200-1.2.shtml as viewed on June 11, 2008
76. Apple Inc., remote Scripting with IFRAME, http://developer.apple.com/internet/webcontent/iframe.html as viewed on June 11, 2008
77. W3Schools, The XMLHttpRequest Object, http://www.w3schools.com/Xml/xml_http.asp as viewed on 2008-5-28
78. Wikimedia Foundation Inc., “Ajax (programming)”, http://en.wikipedia.org/wiki/Ajax_(programming) as viewed on 2008-5-28
79. Garrett J.J., “Ajax: A New Approach to Web Applications”, February 18, 2005, http://www.adaptivepath.com/publications/essays/archives/000385.php as viewed on 2008-5-28
80. Honeywell Limited Australia, “Enterprise Buildings Integrator Configuration and Administration Guide”, Australia, Feb.2002
81. Johnson Controls, “Systems Integration”, http://www.johnsoncontrols.com/publish/us/en/products/building_efficiency/capabilities/systems_integration.html as viewed on 2008-5-28
82. (Honeywell) Alerton, “Programmer’s Guide and Reference, BACtalk Systems”, USA, 2007
83. Echelon Corporation, “LNS DDE Server User's Guide Version 2.01”, CA, USA
84. Microsoft Cooperation, “How To Use QueryPerformanceCounter to Time Code”, http://support.microsoft.com/kb/172338/en-us/ as viewed on 2008-5-28
85. Wikimedia Foundation Inc., “Windows Task Manager”, http://en.wikipedia.org/wiki/Windows_Task_Manager as viewed on 2008-5-28
86. “What is latency - A Word Definition From the Webopedia Computer Dictionary”, http://www.webopedia.com/TERM/l/latency.html as viewed on 2008-5-28
87. ICONICS, Inc. ICONICS OPC Server Simulator, http://www.iconics.com/ as viewed on October 10, 2004
88. Advosol Inc., XMLDA.NET White Paper (2004) http://www.advosol.com/driver.aspx?Topic=WhitePaperXMLDANET as viewed on Oct. 2007
89. Wang S.W., Xu X.H. and Ma Z.J., “Evaluation of Energy Saving Potential of New Buildings in Construction through Intelligent Control”, *SICHUAN - HONG KONG Joint Symposium 2006*, Chengdu, 30th June - 1st July 2006

90. International Business Machines Corp., “Web Services Security”, Updated 01 Mar 2004, http://www-128.ibm.com/developerworks/library/specification/ws-secure/ as viewed on 2008-5-28
91. “Understanding Web Services”, http://www.webopedia.com/DidYouKnow/Computer_Science/2005/web_services.asp as viewed on 2008-5-28
92. Tidwell D., “Web services: The Web's next revolution. 2000”, 1. http://www.cn-java.com/download/book/wsbasics-a4.pdf as viewed on 2008-5-28
93. Microsoft Corporation, “SQL Server 2008 Overview, data platform, store data Microsoft”, http://www.microsoft.com/sqlserver/2008/en/us/default.aspx as viewed on 2008-5-28
94. Sun Microsystems, “MySQL The world's most popular open source database”, http://www.mysql.com/ as viewed on 2008-5-28
95. The MathWorks, Inc., “The MathWorks - MATLAB and Simulink for Technical Computing”, http://www.mathworks.com/
96. Modbus-IDA, http://www.modbus.org/ as viewed on 2008-5-28
97. CAN in Automation (CiA) group, “CAN in Automation (CiA) Controller Area Network (CAN)”, http://www.can-cia.org/ as viewed on 2008-5-28
98. PROFIBUS & PROFINET International (PI) , “PROFIBUS”, http://www.profibus.com/ as viewed on 2008-5-28
99. OPC Foundation, “The OPC Foundation - Dedicated to Interoperability in Automation”, http://www.opcfoundation.org/ as viewed on 2008-5-28
100. LonMark International, “LonMark - Guides & Specifications - Functional Profiles”, http://www.lonmark.org/technical_resources/guidelines/functional_profiles as viewed on 2008-5-28
101. Michael Wetter and Philip Haves, “A MODULAR BUILDING CONTROLS VIRTUAL TEST BED FOR THE INTEGRATION OF HETEROGENEOUS SYSTEMS”, Third National Conference of IBPSA-USA, Berkeley, California, July 30 – August 1, 2008
102. Xu Z. Y., Wang S. W., “Intelligent Building Integration System based on OPC and Web Service MiddlewareTechnology, Intelligent Building & City Information, V1, pp.42-45, Jan. 2007

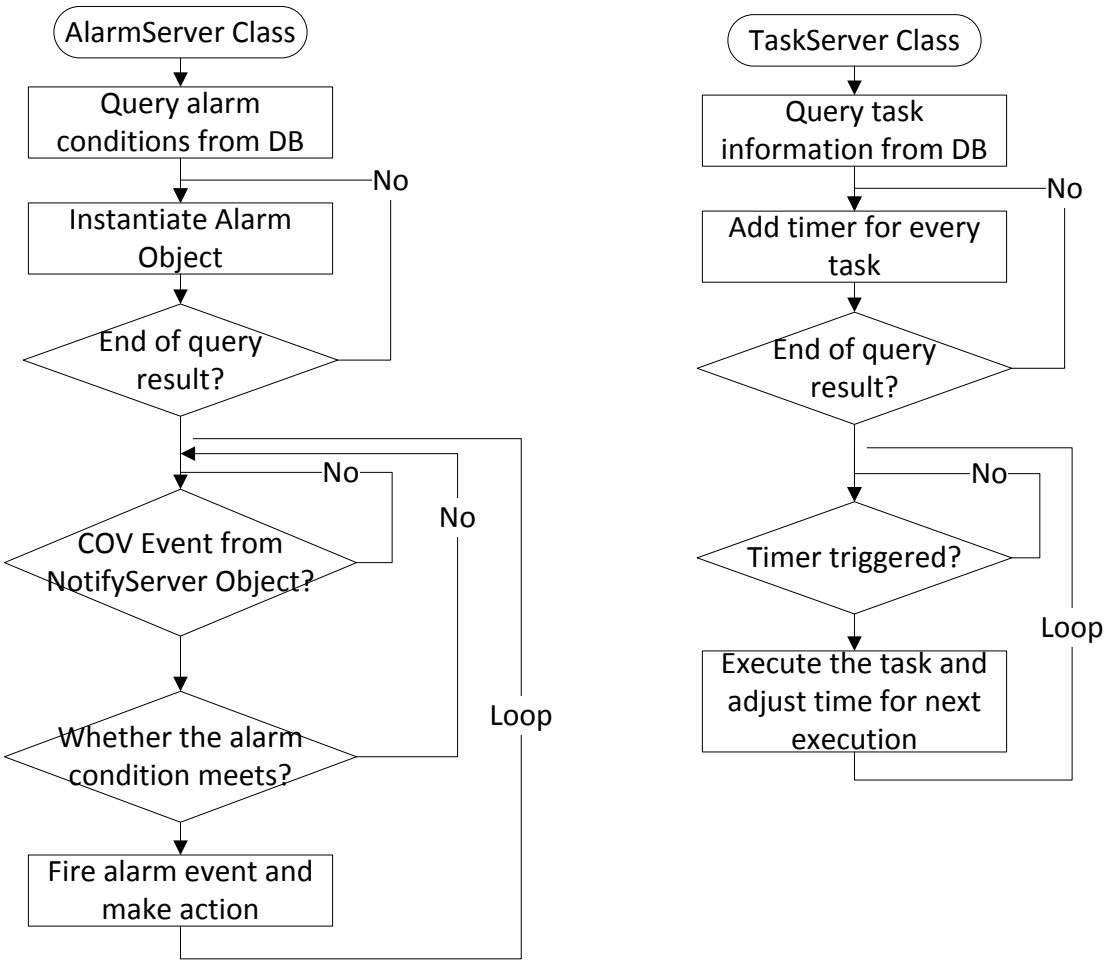
Appendix A - Flow Chart of IBmanager Components



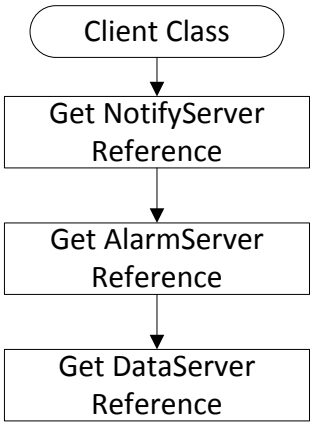
Flow Chart of IBmanager Server Components



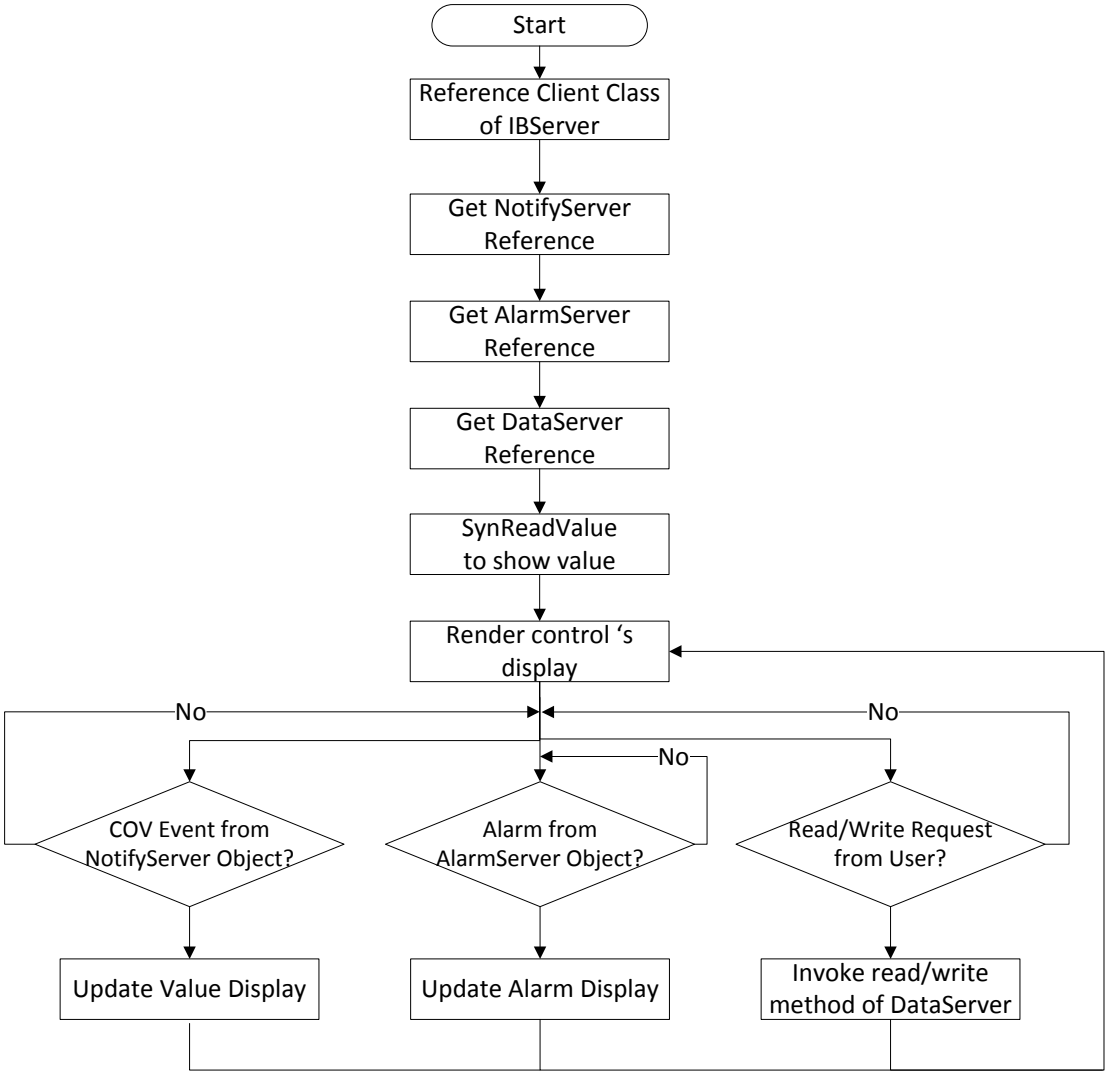
Flow Chart of NotifyServer class and HistoryServer class



Flow Chart of AlarmServer class and TaskServer class



Flow Chart of Client class



Flow Chart of client components

Appendix B - Standard of the Common Interface for DLLs

Properties:

IsConnected

Returns TRUE if the object is currently connected to back-end service, FALSE otherwise.

ItemCount

Returns number of items currently hold in the driver object.

Items

Returns a long string containing item information as follows. The string is multi-line string, i.e. using CR-LF combination to separate each, and every line contains key-value pairs, as shown below.

```
alias=HWtemp;perm=R;...
alias=HWdoor;perm=RW;...
alias=HWcamera;perm=R;...
```

Pre-defined keys are:

- alias - the alias submitted by AddPoint() method.
- perm - the permission actually applied (not requested) to the data point.

Methods:

InitObject(InitString) as boolean

Initialize the driver object.

- InitString (string) - a string of key-value pairs which specify how a specific data source to be connected and described. Return TRUE if the initialization process is successful, FALSE otherwise.
    - OPC: "Node=xxxx;ProgID=xxxxx;"

Connect() as boolean

Make the driver to connect to the back-end system. True if connected, false otherwise.

Disconnect()

Disconnect from the back-end system

Browse([out] RawItemList) as boolean

Returns all accessible data items in a row form.

- RawItemList (string) - [out] this string is ued to store the item list in this format: each item name forms one line, terminated by a CR-LF combination (0x13,0x10).

AddPoint(Alias, ItemName, ReqPermission, Options) as boolean

Add a data point to the object internal container for later access. Return true if item successfully added, false otherwise.

- Alias (string) - the alias representation for a data point in IB Manager
- ItemName (string) - the original item name by the back-end environment. e.g. for OPC, it would be OPC item name.
- ReqPermission (string) - the permission to the data point requested. Include "R" to mean Read, "W" to mean write. E.g., it can be "RW" or "R".
- Options (string) - string of key-value pairs to supply additional options to the object.

SyncRead(Alias, [out] Value, [out] Timestamp) as boolean

Read an alias (mapped with data point) in synchronous way. Return true if the operation succeeded, false otherwise.

- Alias (string) - the alias
- Value (variant) - [out] the value read, if succeeded.
- Timestamp (long) - [out] the timestamp of the operation, value is the number of seconds since 01-Jan-1970, 0:00:00.

SyncWrite(Alias, Value) as boolean

Write a value to a data point. Return true if successful, false otherwise.

- Alias (string) - data point to write to
- Value (variant) - value to write

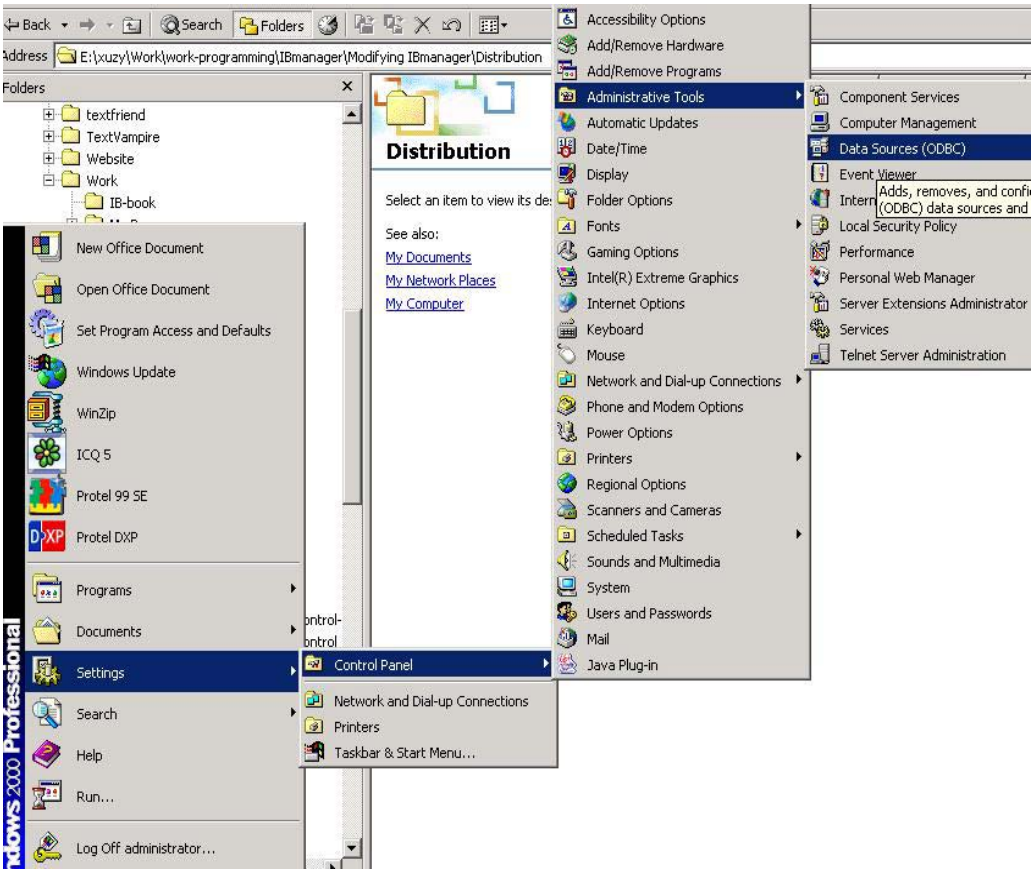
AsyncWrite(Alias, Value) as boolean

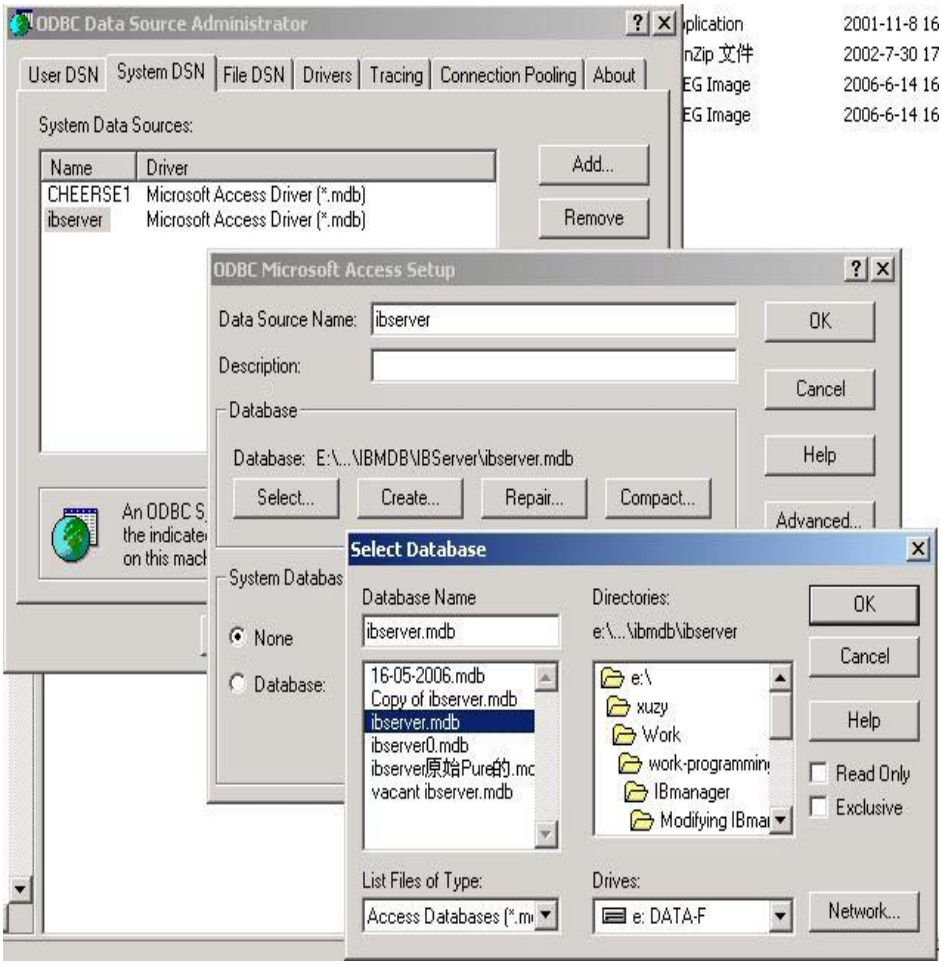
Write a value to a data point, in asynchronous way. Return true if successful, false otherwise. The function will return as soon as possible, not waiting the back-end operation to complete. This specification does not define how it notify the caller the completion of the write.

- Alias (string) - data point to write to
- Value (variant) - value to write

Appendix C – Installation and Operation Manual of IBmanager

1. Install OPC_DA20_Components.exe for OPC access;
2. Install OPC Simulator, run setup.exe in iconics OPC-server.zip (OPC Simulator provided as a data source to test IBmanager system when no real BAS system/device is connected);
3. Install Matlab runtime components and DLL by running MCRInstaller.exe;
4. Install IBServer, *then add "IUSR_***" user to have write & read permission on the IBServer installation folder*; it can solve the error "[Microsoft][ODBC Microsoft Access Driver] Could not use '(unknown)'; file already in use.";
5. Make a system DSN with name "ibserver", which refers to IBServer.mdb in the ibserver installation folder;


Back
Search
Folders
Address
E:\xuzy\Work\work-programming\IBmanager\Modifying IBmanager\Distribution
Folders
textfriend
TextVampire
Website
Work
IB-book
Distribution
Select an item to view its de
See also:
My Documents
My Network Places
My Computer
New Office Document
Open Office Document
Set Program Access and Defaults
Windows Update
WinZip
ICQ 5
Protel 99 SE
Protel DXP
Programs
Documents
Settings
Search
Help
Run...
Log Off administrator...
Control Panel
Network and Dial-up Connections
Printers
Taskbar & Start Menu...
Accessibility Options
Add/Remove Hardware
Add/Remove Programs
Administrative Tools
Automatic Updates
Date/Time
Display
Folder Options
Fonts
Gaming Options
Intel(R) Extreme Graphics
Internet Options
Keyboard
Mouse
Network and Dial-up Connections
Phone and Modem Options
Power Options
Printers
Regional Options
Scanners and Cameras
Scheduled Tasks
Sounds and Multimedia
System
Users and Passwords
Mail
Java Plug-in
Component Services
Computer Management
Data Sources (ODBC)
Event Viewer
Local Security Policy
Performance
Personal Web Manager
Server Extensions Administrator
Services
Telnet Server Administration
Adds, removes, and confi
(ODBC) data sources and


ODBC Data Source Administrator
User DSN
System DSN
File DSN
Drivers
Tracing
Connection Pooling
About
System Data Sources:
Name
Driver
CHEERSE1
Microsoft Access Driver (*.mdb)
ibserver
Microsoft Access Driver (*.mdb)
Add...
Remove
ODBC Microsoft Access Setup
Data Source Name:
ibserver
Description:
Database
Database: E:\...\IBMDB\IBServer\ibserver.mdb
Select...
Create...
Repair...
Compact...
OK
Cancel
Help
Advanced...
System Databas
None
Database:
Select Database
Database Name
ibserver.mdb
16-05-2006.mdb
Copy of ibserver.mdb
ibserver.mdb
ibserver0.mdb
ibserver原始Pure的.mc
vacant ibserver.mdb
Directories:
e:\...\ibmdb\ibserver
e:\
xuzy
Work
work-programmin
IBmanager
Modifying IBmar
OK
Cancel
Help
Read Only
Exclusive
List Files of Type:
Access Databases (*.m
Drives:
e: DATA-F
Network...
plication 2001-11-8 16
nZip 文件 2002-7-30 17
EG Image 2006-6-14 16
EG Image 2006-6-14 16

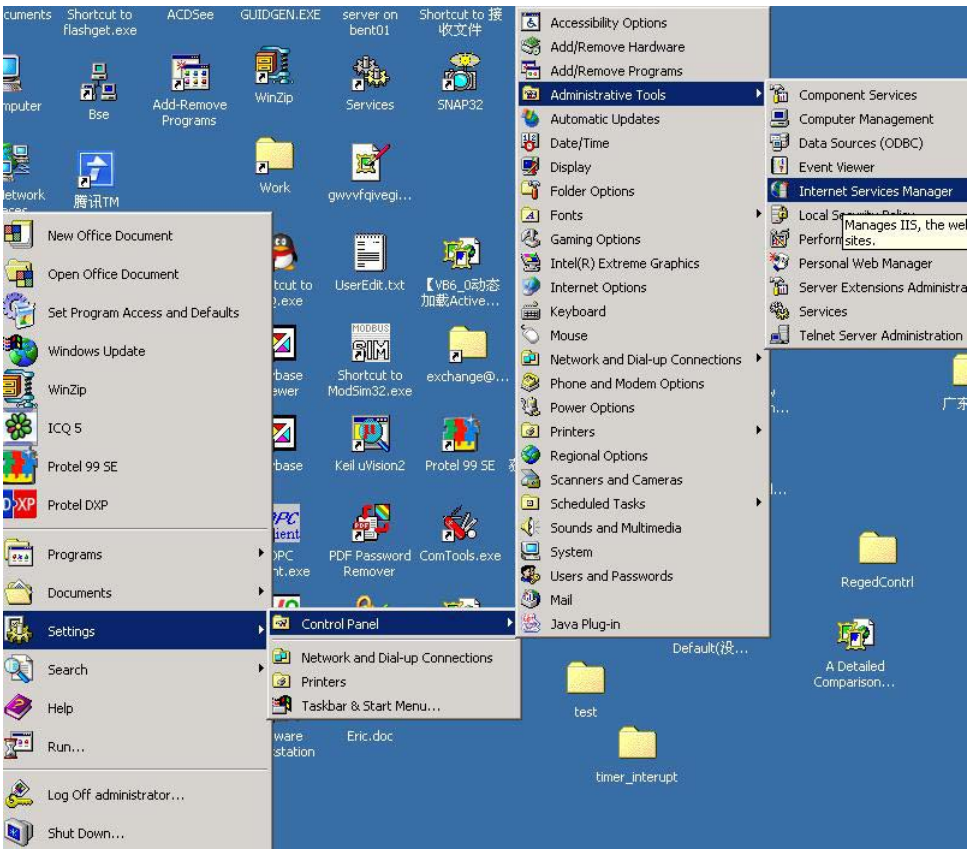
6. Install IBcontrol (include HMI IBcontrol, OPC driver DLL, BACnet driver DLL)

Note: If you want to input data from a file, you should put the data file in the folder the same with IBMDrvFilePoint.dll

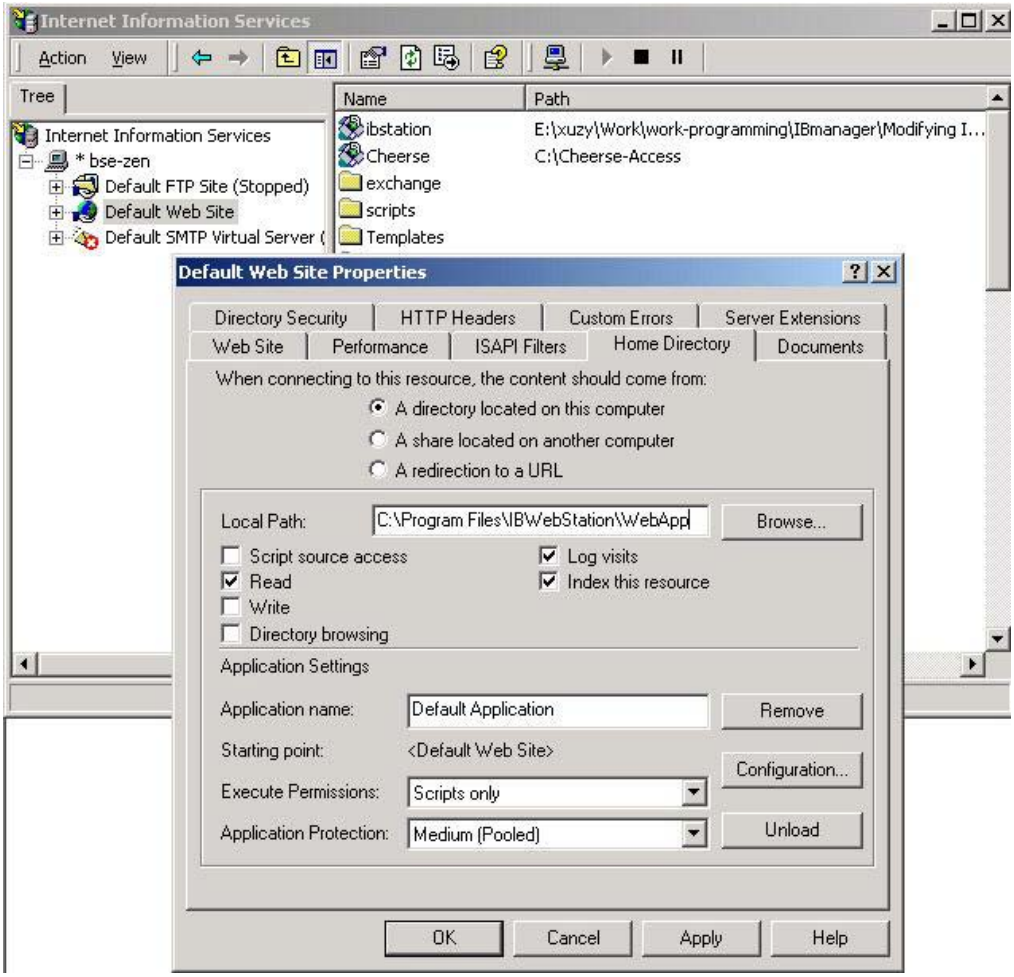
7. Install IBSation (Do not change the default installation folder - C:\Program Files\IBWebStation\)

8. Make a virtual folder refer to the asp file folder in the IBStation folder in the Internet Services Manager

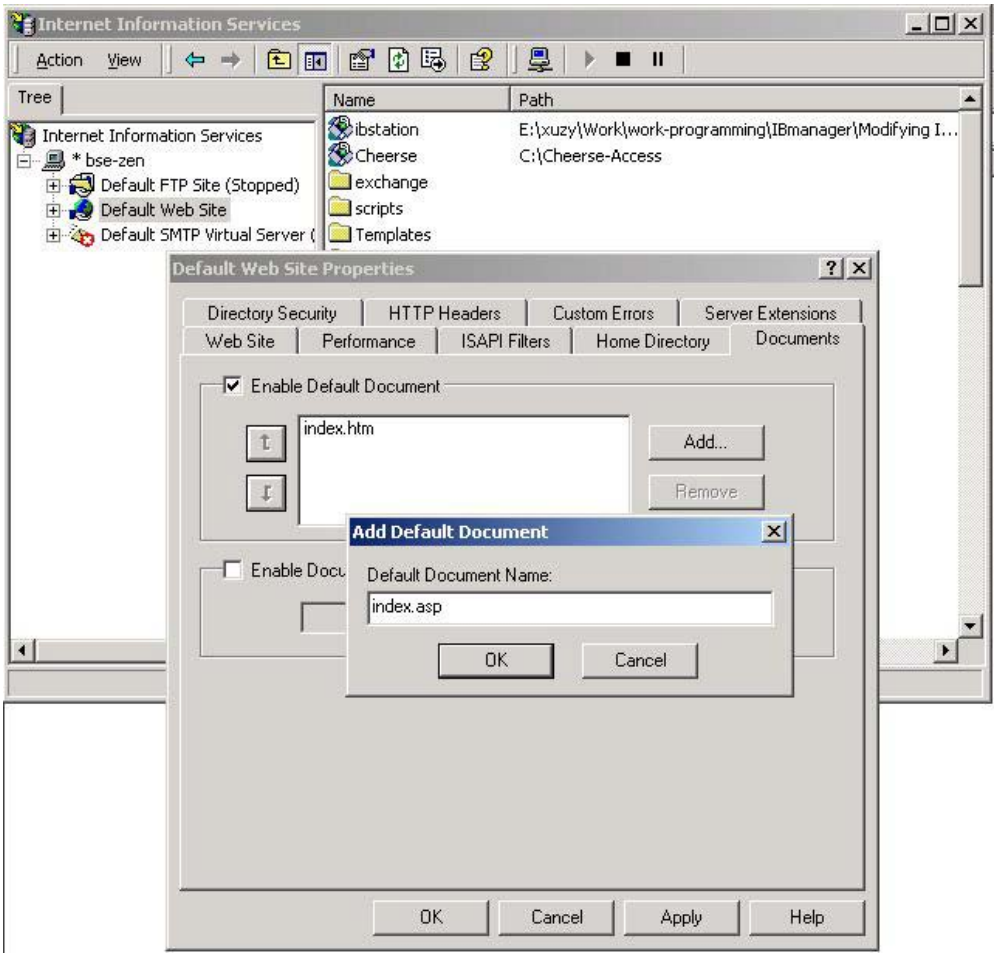
   Step1: Open the IIS Manager Window



Step 2: Locate the “Local Path” for the website

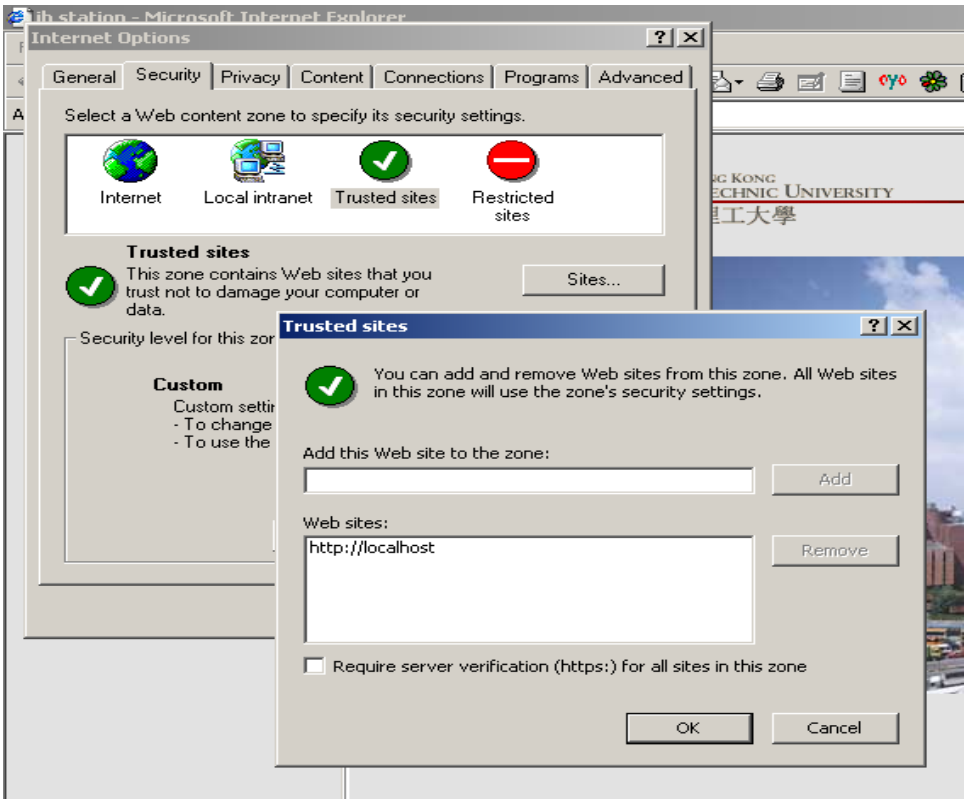


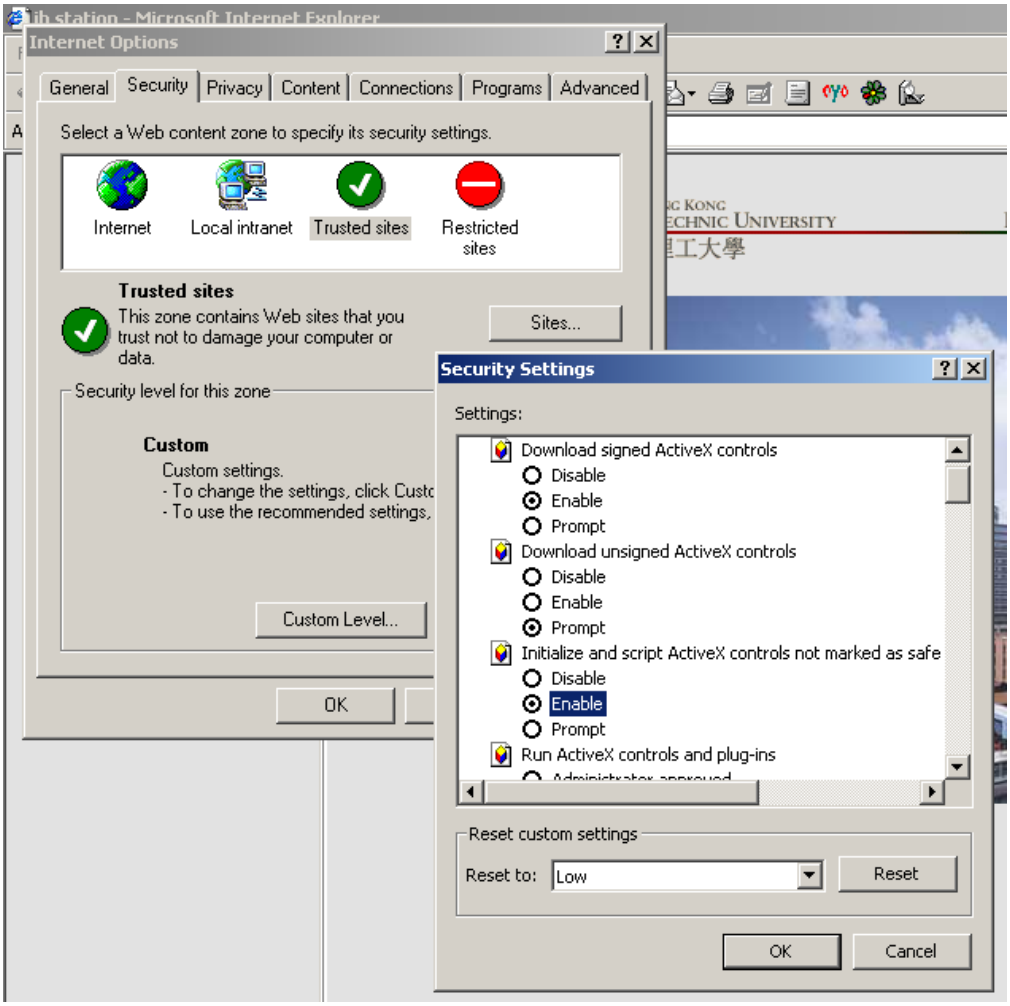
Step 3: Add index.asp to default document



9. Reboot the PC;

10. Then browse the index.asp in the virtual folder from the IE browser (http://localhost/index.asp). It should automatically start IBServer. If it pops up a security alarming message, configure the security setup of IE as below:


Internet Options
General
Security
Privacy
Content
Connections
Programs
Advanced
Select a Web content zone to specify its security settings.
Internet
Local intranet
Trusted sites
Restricted sites
Trusted sites
This zone contains Web sites that you trust not to damage your computer or data.
Sites...
Security level for this zor
Custom
Custom settir
- To change
- To use the
Trusted sites
You can add and remove Web sites from this zone. All Web sites in this zone will use the zone's security settings.
Add this Web site to the zone:
Add
Web sites:
http://localhost
Remove
Require server verification (https:) for all sites in this zone
OK
Cancel

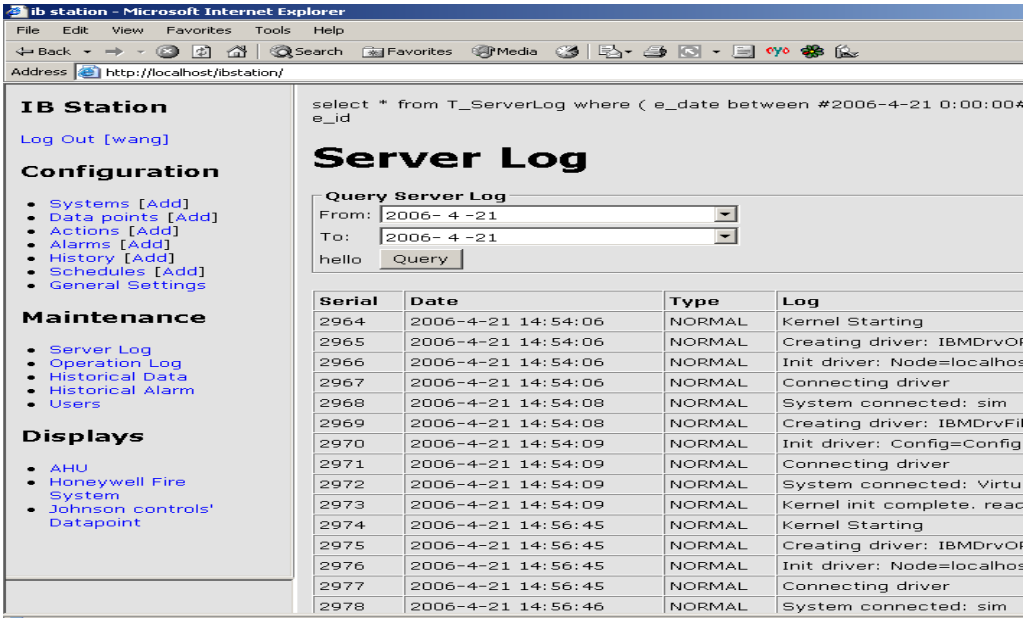


11.log in the web station, username:wang, password:wang

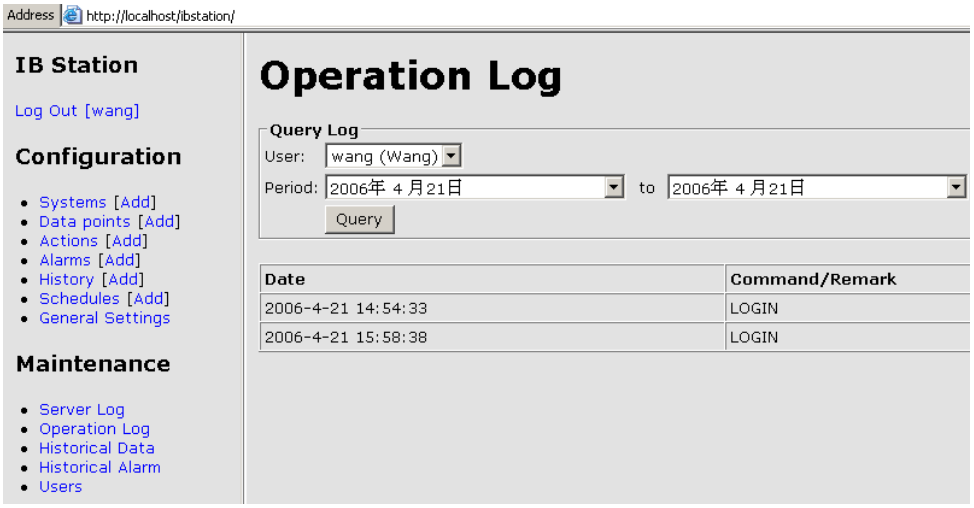
12.Then you can do the operations below:

Maintenance:

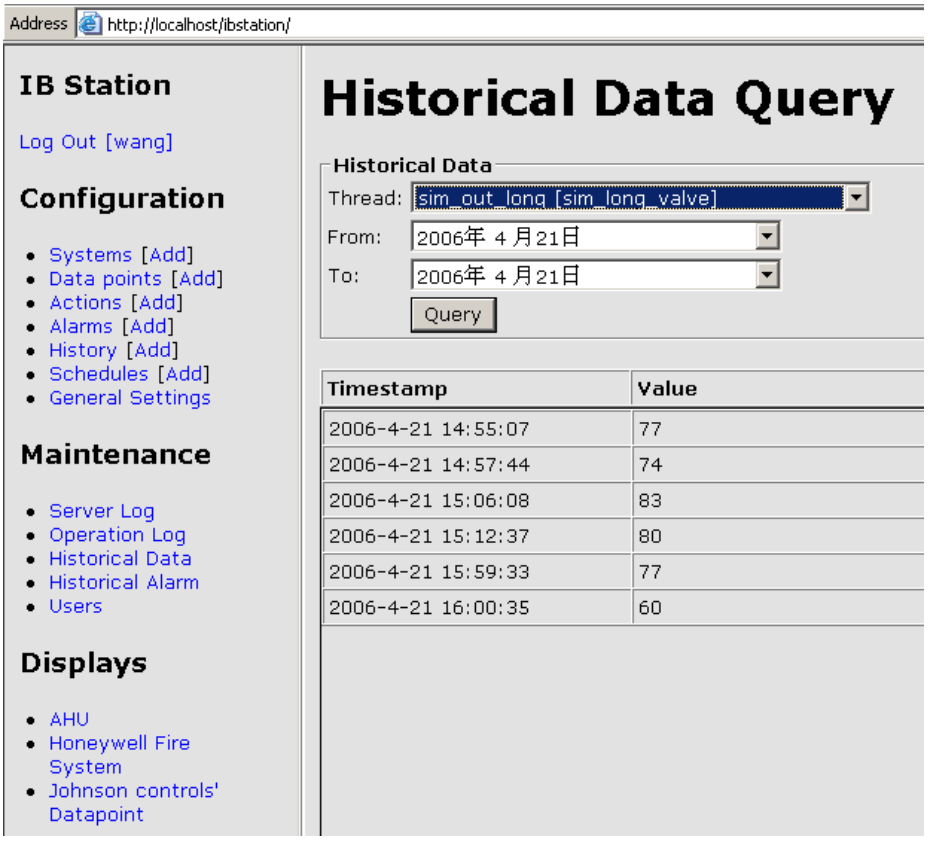
(1) View Server Log



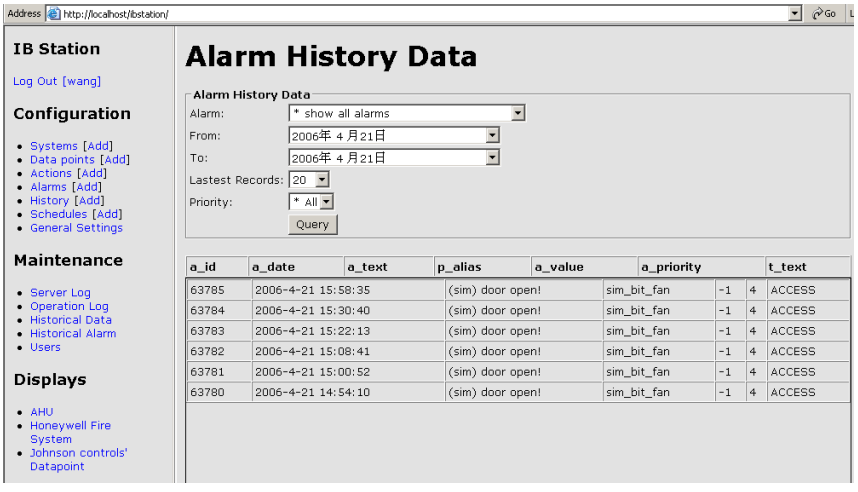
(2) View Operation Log



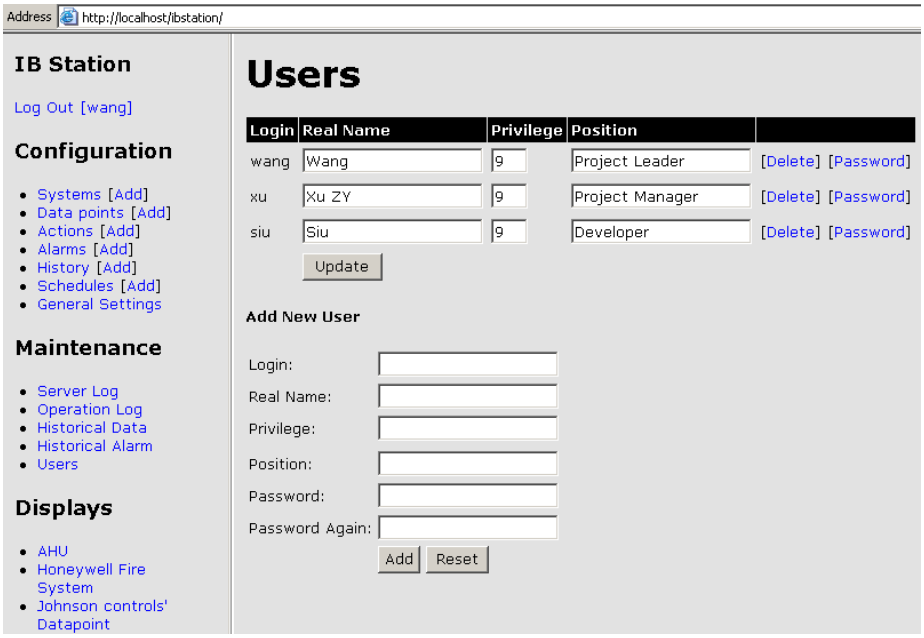
(3) View Historical Data



(4) View Alarm History

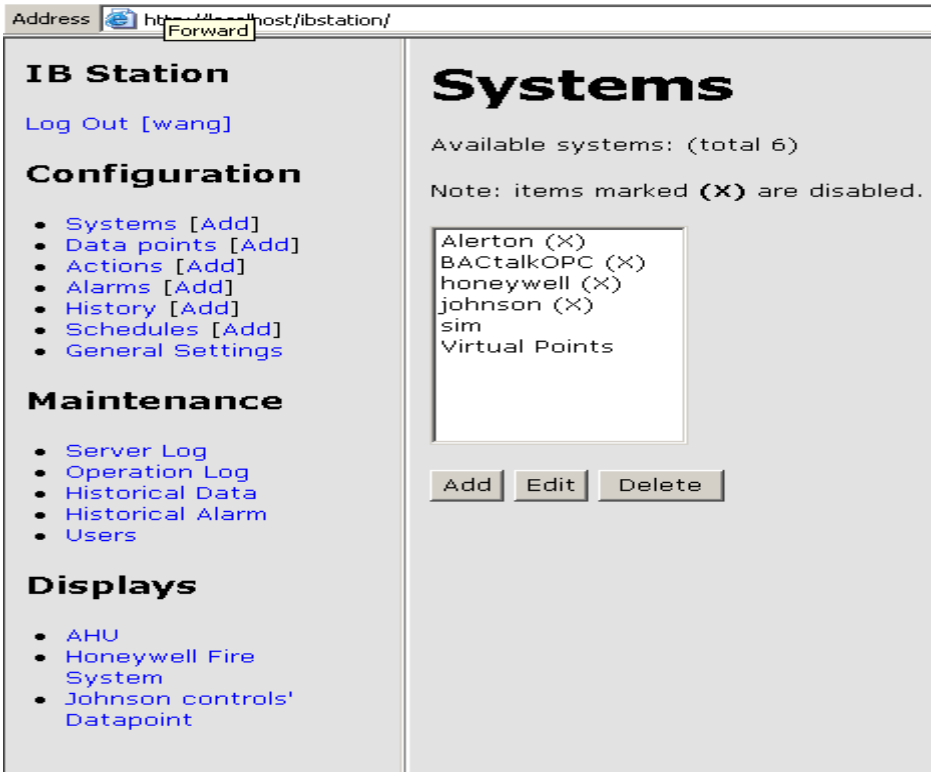


(5) Maintain Users

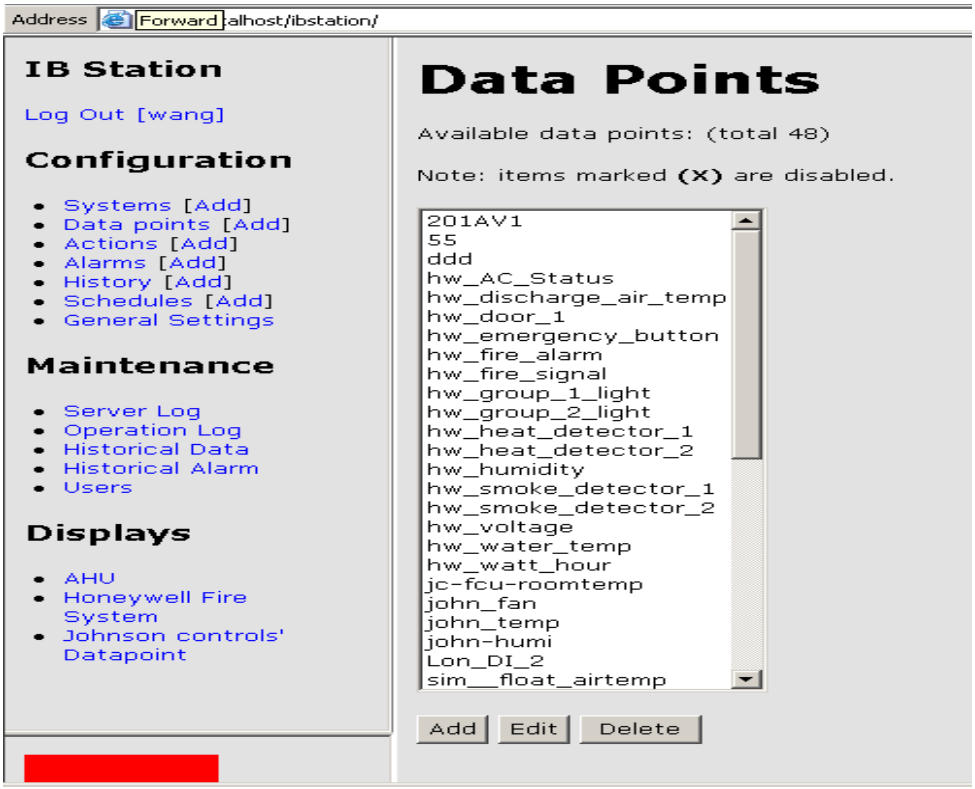


Configuration:

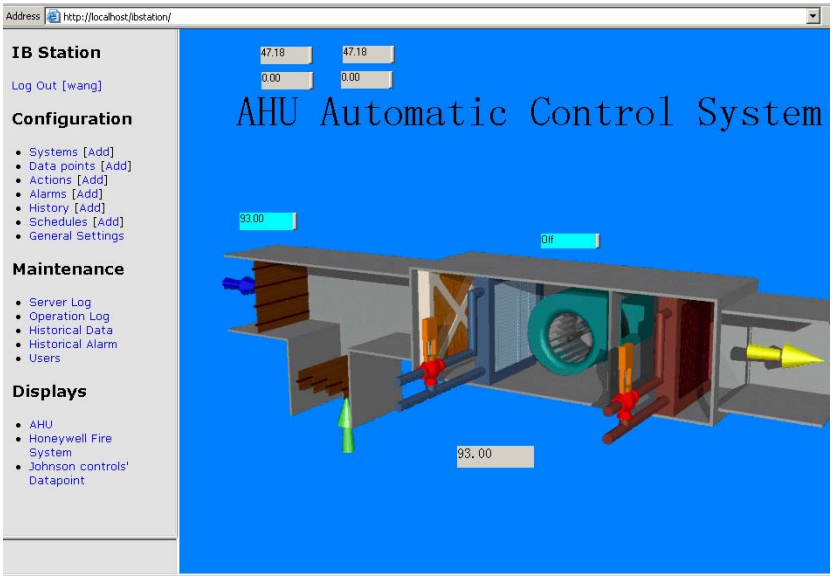
(1) Configure Drivers

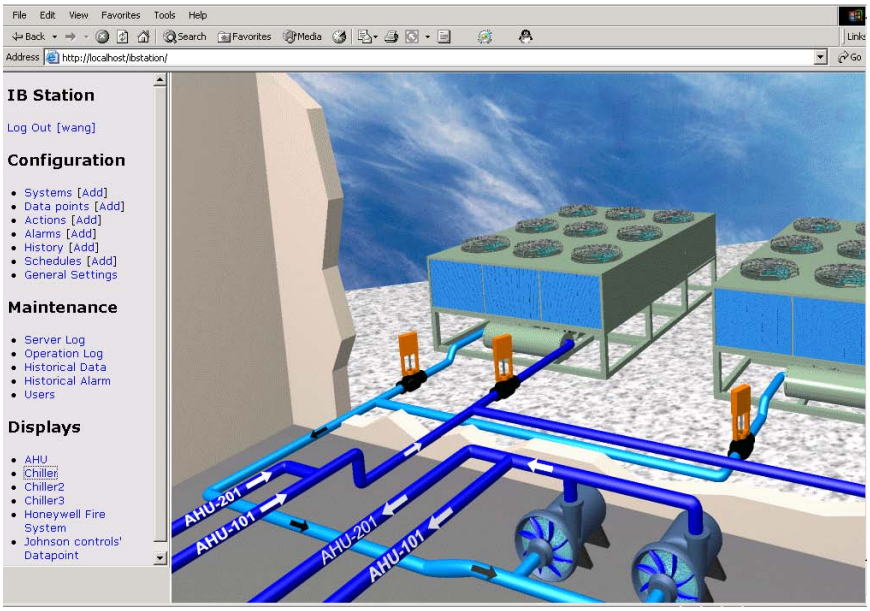


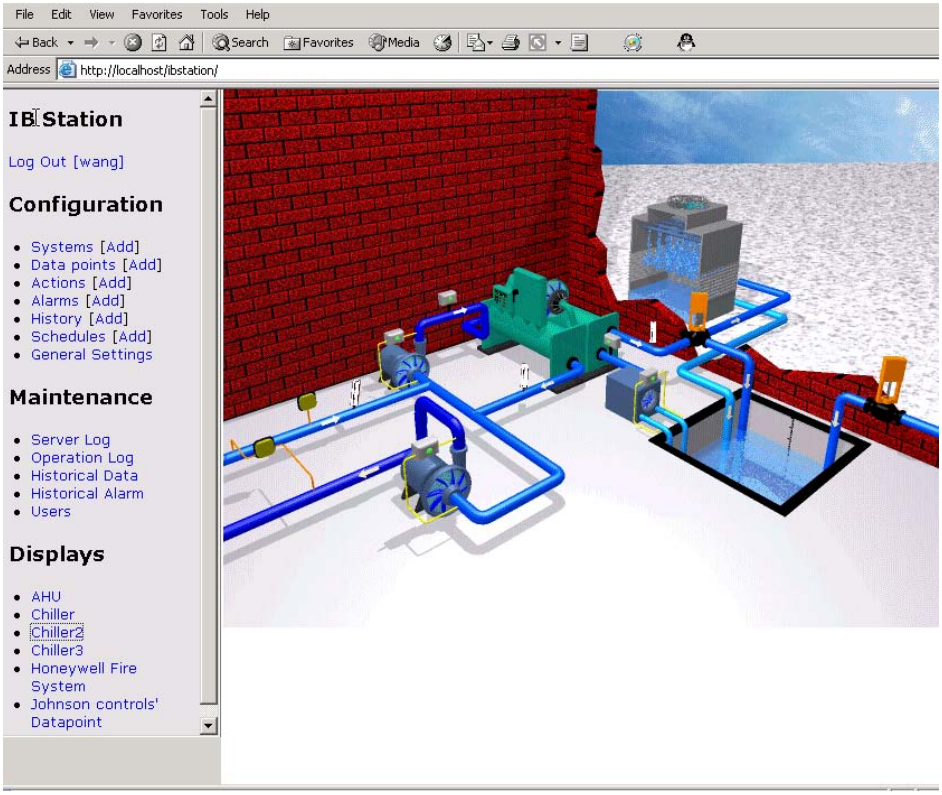
(2) Configure Data Points



Displays:







The pictures used are placed in C:\Program Files\IBWebStation\WebApp\Images folder.

End.