Database Supported BACnet Data Acquisition System For Building Energy Diagnostics

Zhengwei Li
Ph.D Student
Georgia Institute of Technology
Atlanta, GA

Godfried Augenbroe
Professor
Georgia Institute of Technology
Atlanta, GA

Bing Dong
Senior Research Scientist
United Technologies Research Center
East Hartford, CT

Zheng O'Neill
Principal Investigator
United Technologies Research Center
East Hartford, CT

ABSTRACT

This paper reports a tool that can be used to acquire and store the BACnet (A Data Communication Protocol for Building Automation and Control Networks) data for the purpose of building energy system Fault Detection and Diagnostics (FDD). Building Automation Control (BAC) systems have become a common practice in recently constructed buildings in the United States. Although building operational data could readily be collected for various analysis purposes, there is still a debate in building community which or what FDD method is better in terms of performance matrix, such as false alarm rate and training data requirement, etc. Therefore, from the user's perspective, it is potentially beneficial to try out different FDD methods before the deployment, or even develop a dedicated FDD method in a specific case. This is the motivation for development of the BACnet data storage system discussed in this paper, which could then be used together with BACnet data acquisition module in an open source Building Control Virtual Test Bed (BCVTB) environment [2].

This paper discusses (1) Relational database schema development for the purpose of storing building operational data and FDD analysis data (2) Development of the connector in BCVTB that enables the transition from the BACnet module to the database module and (3)Testing of the integrated system in a real building. The relational database is intended to be general and detailed enough so that it can be applied to different buildings and projects with various complexity without any major structure change. The BACnet-reader to database connector enables seamless data flow from commercial BACnet system to user's customized workstation. The integrated system enables users to analyze building operational data in an effective and efficient way, which helps achieve automated FDD in buildings.

INTRODUCTION

BACnet (A Data Communication Protocol for Building Automation and Control Networks) system [1] has become more and more popular in commercial buildings in the United States. Along with its popularity, the number of buildings that have real time indoor information and system operational information available is bumping up, which makes more advanced control (such as model predictive control), and more sophisticated FDD method practically implementable.

BACnet systems typically have two interfaces that user can interact with. On the lower level, a local controller has a small processing unit, upon which user can write simple algorithms for various purposes. On the higher level, a central Energy Management System (EMS) in most cases is a server level computer, on which user can install software to implement intended functions.

Although a user can choose either one of these two interfaces to work with, typically, working with a local controller is not very convenient, and there are some unavoidable restrictions. One of the restrictions is that since the computational power is limited, the algorithm could not be too complex. Compared to local controller, working with a BACnet system at the central EMS level is much more convenient. The user interface is the same as the computer used in home and office, the computational power is much stronger, and it has access to more points in the BACnet system.

To work with central EMS, a convenient choice is using the software provided by BACnet system vendors. But the functionalities supported by this kind of tool are often very limited. The mostly common functionalities are browsing, logging and trending. If user has a self-developed Fault Detection and Diagnostics (FDD) algorithm, very likely it can only be used off-line with the log data.

To explore the flexibility of interacting with BACnet system in real time, there have been certain initiatives going on. A very influential BACnet reading/writing package is BACnet stack [3]. It is an open source BACnet protocol stack for embedded system. To provide the functionality of batch transaction, and connecting BACnet system with dynamic simulation software, Lawrence Berkeley National Lab (LBNL) has developed a BACnet module in their co-simulation software – Building Control Virtual Test Bed (BCVTB) [4].

The work introduced in this paper can be regarded as an extension of the work done in LBNL. Although BCVTB enables the real time processing of BACnet system data points, it throws away the data after transaction is finished. If user wants to see the history record of the points, he/she will have to go back to the proprietary software provided by the vendor. To enable the storage of BACnet data points, the authors have developed a relational database system that serve as the medium to store the data, and a set of actors that connect BCVTB to the database. As a result of this work, a fully functional BACnet data acquisition and storage system is successfully installed and under service.

The following content is divided into two sections. The first section is about the development of a database, with the vision of seamless connection with Building Informational Modeling (BIM) compatible files. The second section introduces the development of the actors connecting BCVTB to database, in which an example that combines BACnet Reader, transition actor and database actor will be given.

DATABASE DEVELOPMENT

The database development effort went through four steps: (1) functional specification (2) software selection (3) schema development (4) database establishment. They will be described in the following sequentially.

Functional Specification

To achieve the objective of serving as an information exchange repository for multiple function modules, including simulation module, FDD module and real time data acquisition module, as well as providing an interface to connect with a Building Information Model (BIM) file (such as IFC file), following functional requirement specifications for the database are identified:

- Information storage requirements
  - Supports operational, simulation, FDD modules
  - Supports storage of major building information in BIM software (such as Revit) generated file
- Information integrity requirements
  - Protect data from unauthorized transaction
- Information retrieval requirements
  - Easiness to retrieve desired data based on certain rules
- Performance requirements
  - Performs read/write transaction within reasonable amount of time
- Data filtering requirements
  - Allows user to do fairly complex data filtering algorithms

Software Selection

To stand in line with the open source BCVTB platform, the chosen database platform is also expected to be open source or free.

Currently, there are two major free database systems on the market: MySQL [5] and PostgreSQL [6]. Just to compare their performance at a high level, MySQL is better in transaction speed and stability, PostgreSQL has more complete function support but is slower. Although MySQL maybe faster in transactions, pilot test shows that PostgreSQL has a fairly reasonable speed. On the first four requirements, MySQL performs slightly better than PostgreSQL, but on the last requirement, PostgreSQL has better support than MySQL.

Therefore, PostgreSQL is chosen over MySQL because of its better functional support.

Schema Development

Corresponding to the functional requirements, the schema can be divided into four components: (1) static Information component (e.g., building geometry, mechanical equipment schedules etc.), used to store static information in IFC compliant files and required by other modules (2) real time operational information component, used to store operational information from BACnet data acquisition system (3) simulation information component, used to store simulation outputs from simulation module (4) FDD information component, used to store FDD inputs and outputs from the FDD module.

1. Static Information

The static information component includes thermal property related building envelope information, building occupancy/schedule information and building system information. Building system includes lighting system, power system, and HVAC system, etc.

Since thermal simulation is the purpose of storing building envelope information, the schema entities follow the traditional building energy simulation modeling naming convention. The physical elements are categorized at three levels: building level, zone level and room level. At building level, the location and weather information are stored. At zone and room levels, the walls, windows and material properties information are stored. Figure 1 shows the entities at zone and building level, Figure 2 shows the entities at room level.

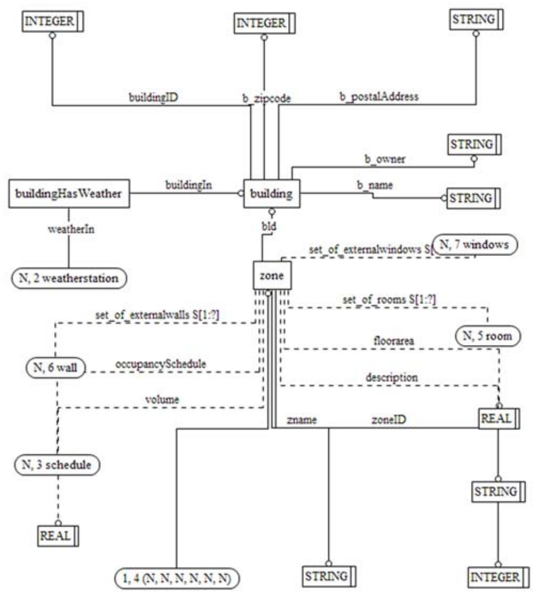


Fig1 Building and Zone

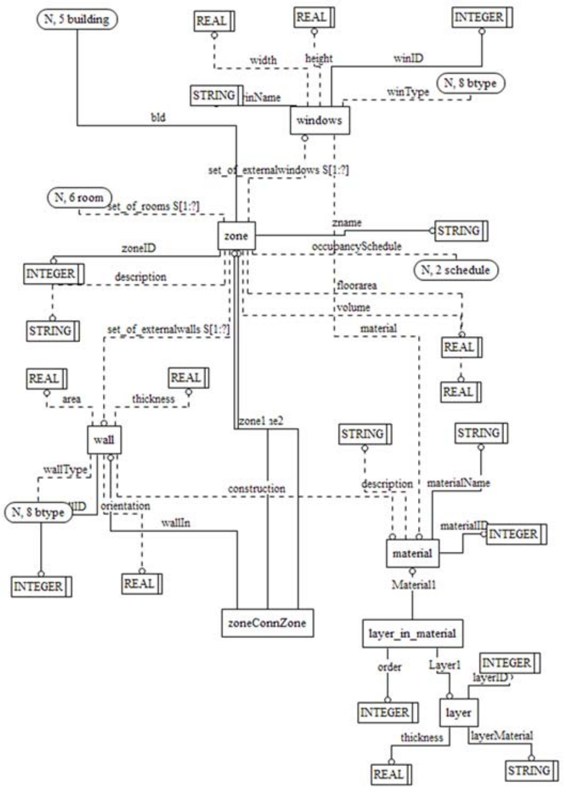


Fig2 Zone, Room, Wall, Window and Material

To perform building energy simulation, property information about building system (HVAC system, renewable energy system, etc.) are necessary. In the schema, there is a top level entity ‘equipment’ that generalizes all the equipment. All equipment can belong to either one of these four categories: Primary System, Secondary System, Terminal System and Component. The top level equipment entity is shown in figure 3. Primary system, secondary system, terminal system and component entities are shown in figure 4, 5 respectively.

To store the performance parameter for equipment at component level, entity “parameter” and “performance” are created, shown in figure 6 and 7 respectively.

The following example (Fig 8) shows the use of parameter and performance for a specific component – pump.

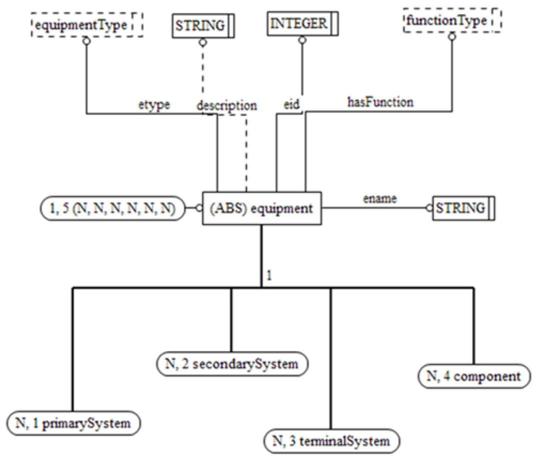


Fig3 Equipment

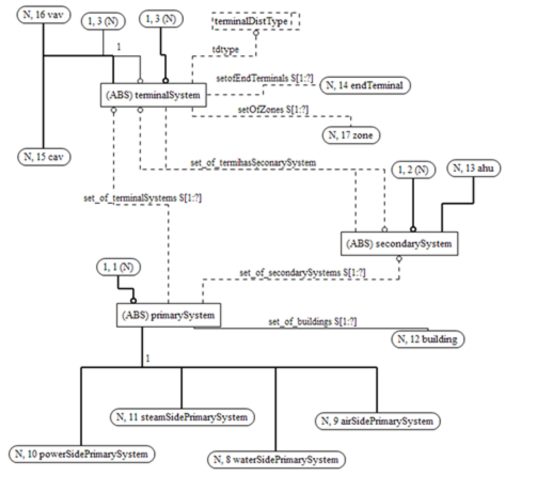


Fig4 Primary, Secondary and Terminal System

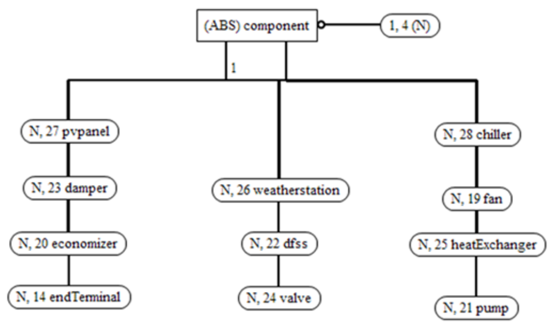


Fig5 Component

2. Operational Information

Operational information comes from a Building Energy Management System, including (1) Sensor data that reflect current conditions in building or systems. (2) Signals, which are generated by local controllers, and (3) Control set points, which are set by occupants or in central EMS. Therefore, in the conceptual level, all operational data are divided into sensors, control signals and set points. At data level, each operational data is associated with a specific object (building, zone or equipment), with name, description and meaningful physical definitions, as shown in Figure 9.

To facilitate the data filtering transaction, all operational data are further separated based on the physical definitions (temperature, velocity, volume flow rate, etc.). The sensor schema developed based on this categorization can be seen in Figure 10. To store the dynamic value of a sensor, timestamp is used as attribute at the smallest category level. As an example, Figure 11 shows the schema for temperature sensor.

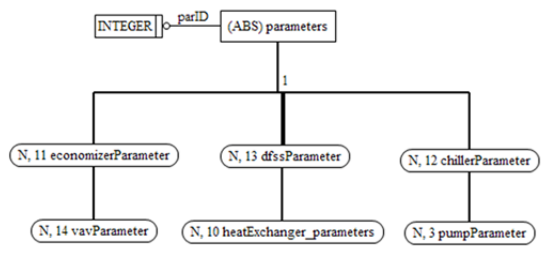


Fig6 Parameter

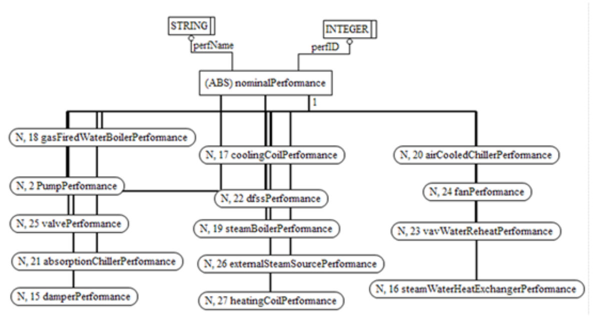

STRING
INTEGER
perfName
perfID
(ABS) nominalPerformance
1
N, 18 gasFiredWaterBoilerPerformance
N, 17 coolingCoilPerformance
N, 20 airCooledChillerPerformance
N, 2 PumpPerformance
N, 22 dfssPerformance
N, 24 fanPerformance
N, 25 valvePerformance
N, 19 steamBoilerPerformance
N, 23 vavWaterReheatPerformance
N, 21 absorptionChillerPerformance
N, 26 externalSteamSourcePerformance
N, 16 steamWaterHeatExchangerPerformance
N, 15 damperPerformance
N, 27 heatingCoilPerformance

Fig7 Performance

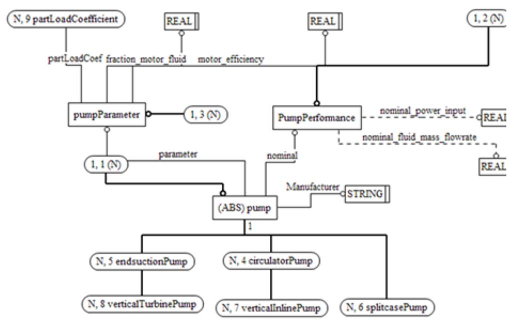


Fig8 Pump

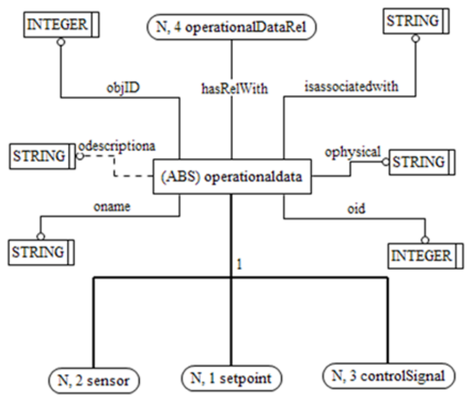


Fig9 Operational Data

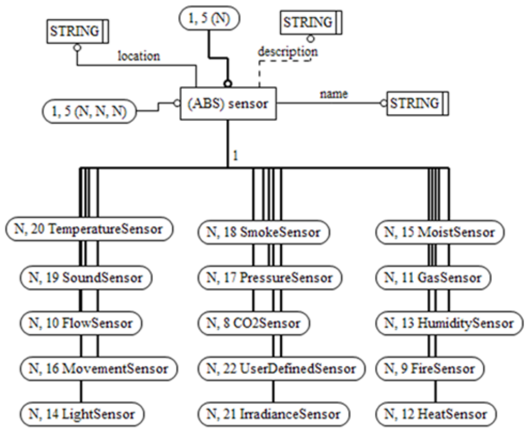


Fig10 Sensor

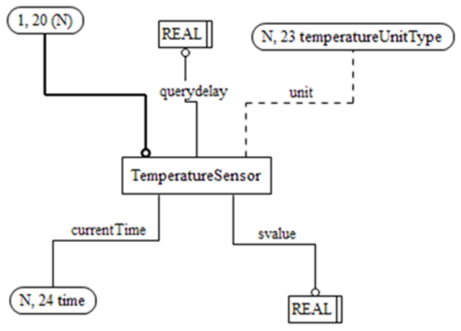


Fig11 Temperature Sensor

3. Simulation Information

The simulation module reads inputs (typically static information) and generates output (typically dynamic information). The simulation inputs are stored with entities defined in building static information. To store the dynamic output of the simulation, a new entity 'simulationdata' has been created, which stores the static information of the data. To store the dynamic output, this entity is further divided into small physics based categories, such as temperature, flow rate, etc. The 'simulationdata' entity is shown in Figure 12, as an example, the 'temperaturesim' entity is shown in Figure 13.

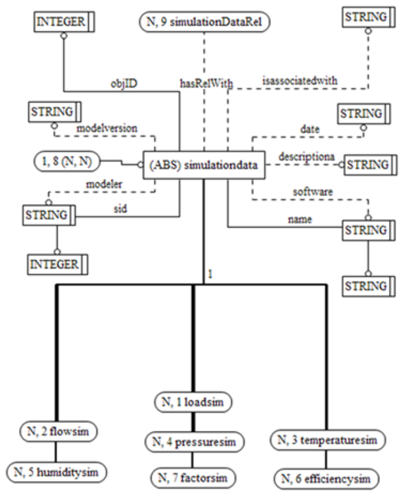


Fig12 Simulationdata

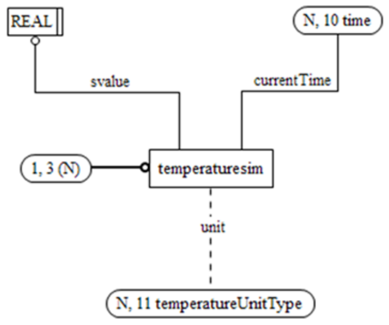


Fig13 Temperaturesim

4. FDD Information

Depending on the detailed FDD method, a FDD module may have different inputs and outputs. Typically operational data and simulation data are the inputs for the FDD module. To map a particular FDD data point to corresponding operational and simulation records, a new entity 'discretedata' has been created. To store the dynamic FDD output, 'dynamicDiscreteData' entity has been created.

Database Establishment

Based on the schema introduced above, a database that includes 160 tables has been developed in PostgreSQL platform. Since this database is general in terms of semantic relationship, it can be used in multiple projects with minor modification.

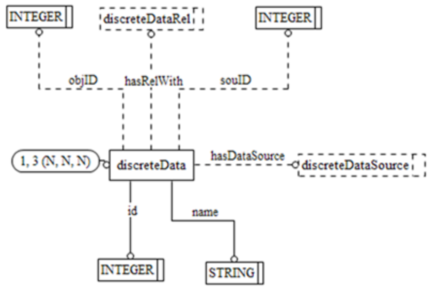


Fig14 DiscreteData

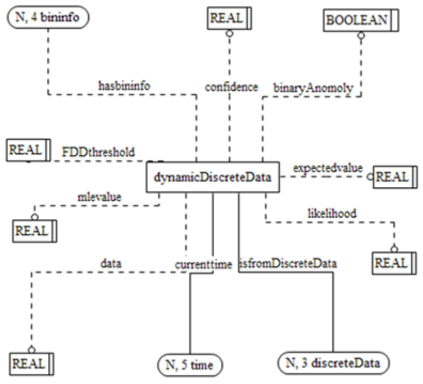


Fig15 dynamicDiscreteData

CONNECT DATABASE TO BCVTB

Introduction of BACnet Reader in BCVTB

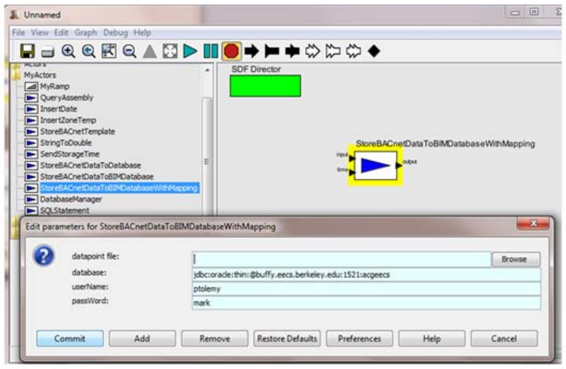
To read data from a BACnet system, the database developed above has to be connected to a BACnet data acquisition system. BCVTB [2] is used in this study. It is a tool developed in Lawrence Berkeley National Lab (LBNL), on the basis of Ptolemy environment [7]. It has a module that leverages open source BACnet stack library [3] to read from and write to BACnet system. It has been shown that BCVTB is a reliable tool in BACnet data acquisition [4].

In BCVTB GUI, there is a BACnet reader actor available on the actor panel. To use it, a user just needs to specify a XML configuration file which is composed of BACnet point list in a predefined format. After making sure the Ethernet connection between computer and the BACnet hardware system is successfully established, and defining the acquisition period and frequency, the user just need to clicks a 'start' button, then at each time step, all the BACnet point data will be read in to BCVTB.

BACnet-Database Connection Actor

BCVTB is shipped with several database actors that can perform simple database transaction functions. Different actors have slightly different functions. Typically a user needs the 'DatabaseManager' actor to establish the connection between BCVTB and the database. The 'DatabaseSelect' actor is used to query the database. The 'DatabaseInsert' actor can be used to insert record to database. The 'SQLStatement' actor can be used to execute any valid SQL statements. The 'SQLStatement' is more flexible than the 'DatabaseInsert' because the table to which records are inserted to can be changed.

To convert the raw data into SQL statements that can then be sent to the 'SQLStatement' actor, an connection actor has been developed, which can been seen in Fig 16.



Edit parameters for StoreBACnetDataToBIMDatabaseWithMapping

datapoint file:
database: jdbc:oracle:thin:@buffy.eecs.berkeley.edu:1521:acgeecs
userName: ptolemy
passWord: mark

Fig16 BCVTB-Database Transition Actor

This transition actor has two input ports: 'input' port and 'time' port. The 'input' port connects to the raw data array, the 'time' port connects to an artificial computer time. The actor has one output port, from which sql commands go to database actors to get executed.

This transition actor requires a data point description file in csv file format, which maps each raw data to a specific record in 'operationaldata' table.

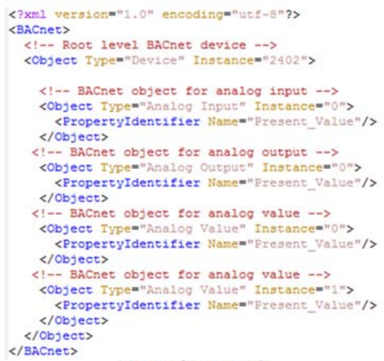
A Working Example

In this simple example, a control panel manufactured by AutomatedLogic Corporation is connected to a Windows 7 computer. In the control panel there are some pre-programmed BACnet object data points. Through command line query, it is already known that these points include 'Analog

Input 0', 'Analog Output 0', 'Analog Value 0' and 'Analog Value 1'.

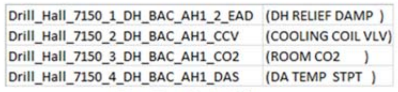
For testing purpose, in the 'operationaldata' table, these four points are defined as 'Air Handling Unit 1 Relief Damper Position', 'Air Handling Unit 1 Cooling Coil Position', 'Room CO2 concentration in Air Handling Unit 1', 'Air Handling Unit 1 Discharge Air Temperature Set point'. These four points are classified into 'position signal', 'position signal', 'CO2 sensor' and 'temperature setpoint' respectively.

To set up this example, a XML configuration file for BACnet objects and data description file need to be prepared. Also the 'operationaldata' table needs to be instantiated. These are shown in fig 17. The example model in BCVTB interface is shown in fig 18. The result of running this example is shown in fig19, the upper part shows the real time data acquired, the lower part shows the response of database transactions.



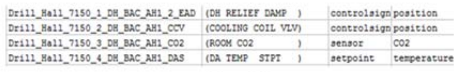
```xml
<?xml version="1.0" encoding="utf-8"?>
<BACnet>
  <!-- Root level BACnet device -->
  <Object Type="Device" Instance="2402">

    <!-- BACnet object for analog input -->
    <Object Type="Analog Input" Instance="0">
      <PropertyIdentifier Name="Present_Value"/>
    </Object>
   <!-- BACnet object for analog output -->
    <Object Type="Analog Output" Instance="0">
      <PropertyIdentifier Name="Present_Value"/>
    </Object>
   <!-- BACnet object for analog value -->
    <Object Type="Analog Value" Instance="0">
      <PropertyIdentifier Name="Present_Value"/>
    </Object>
   <!-- BACnet object for analog value -->
    <Object Type="Analog Value" Instance="1">
      <PropertyIdentifier Name="Present_Value"/>
    </Object>
  </Object>
</BACnet>
```

XML Configuration File



Drill_Hall_7150_1_DH_BAC_AH1_2_EAD	(DH RELIEF DAMP)
Drill_Hall_7150_2_DH_BAC_AH1_CCV	(COOLING COIL VLV)
Drill_Hall_7150_3_DH_BAC_AH1_CO2	(ROOM CO2)
Drill_Hall_7150_4_DH_BAC_AH1_DAS	(DA TEMP STPT)

Data Description File



Drill_Hall_7150_1_DH_BAC_AH1_2_EAD	(DH RELIEF DAMP)	controlsign	position
Drill_Hall_7150_2_DH_BAC_AH1_CCV	(COOLING COIL VLV)	controlsign	position
Drill_Hall_7150_3_DH_BAC_AH1_CO2	(ROOM CO2)	sensor	CO2
Drill_Hall_7150_4_DH_BAC_AH1_DAS	(DA TEMP STPT)	setpoint	temperature

Operationaldata Table

Fig17 Data File Preparation

The integrated system has also been tested in a building located in Chicago. In this building, the EMS system is Siemens Apogee system. A virtual BACnet server has been installed to convert the data points from the proprietary protocol to BACnet protocol, which directly sends data to the system introduced in this paper. In total there are more than two thousand data points, which are read in every five minutes to the system. The data acquisition is done within the order of seconds. The system has been running stably since March, 2011. An on-site system snapshot can be seen in Figure 20.

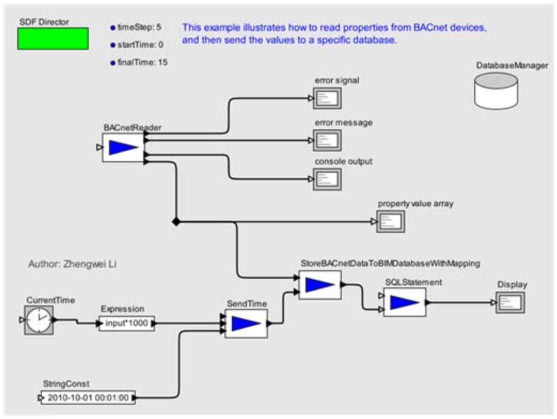
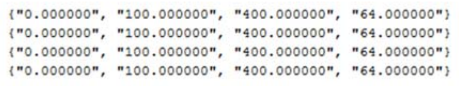
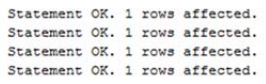


Fig18 Working Example in BCVTB



```
{"0.000000", "100.000000", "400.000000", "64.000000"}
{"0.000000", "100.000000", "400.000000", "64.000000"}
{"0.000000", "100.000000", "400.000000", "64.000000"}
{"0.000000", "100.000000", "400.000000", "64.000000"}
```

BACnet Reader Result



```
Statement OK. 1 rows affected.
Statement OK. 1 rows affected.
Statement OK. 1 rows affected.
Statement OK. 1 rows affected.
```

Database Execution Result

Fig19 Result of Working Example

Fig20 On-site System Snapshot

CONCLUSION

The authors have developed a system that can acquire large amounts of BACnet operational data in real time and store them in a relational database. The data acquisition component is a combination of BACnet reader actor in BCVTB and a BCVTB-to-database connection actor. Because the database supports storage of operational data, simulation data, and FDD data, it can serve as a central data repository in a fully integrated environment.

ACKNOWLEDGEMENT

This work was performed under the project EW-201015 administered by ESTCP (Environmental Security Technology Certification Program) technology program of the Department of Defense. We would like to thank ESTCP program manager, Dr. James Galvin and energy manager at Great Lakes, Mr. Peter Behrens, for their support. We also want to thank Trevor Bailey, Sudha Krishnamurthy in UTRC for their valuable discussions. Views, opinions, and/or findings contained in this paper are those of the authors and should not be construed as an official Department of Defense position or decision unless so designated by other official document.

REFERENCES

1. ASHRAE (2008). The BACnet standards, ANSI/ASHRAE Standards 135-2008.
2. Wetter, M. and P. Haves (2008). "A modular Building Controls Virtual Test Bed for the integration of heterogeneous systems." 3rd SimBuild Conference.
3. BACnet stack – An open source BACnet protocol stack for embedded systems, http://bacnet.sourceforge.net.
4. Nouidui, T. S., M.Wetter, et al. (2011). BACnet and Analog/Digital Interfaces of The Building Controls Virtual Test Bed. IBPSA 2011, Sydney.
5. Eker, J., J. Janneck, et al. (2003). "Taming heterogeneity - the Ptolemy approach." Proceedings of the IEEE 91(1): 18.
6. MySQL community server, http://dev.mysql.com/downloads/mysql/5.5.html
7. PostgreSQL, http://www.postgresql.org