

# Implementation of APIs in Building Automation Systems

**ATN Automation**

Markus Nygård

Bachelor's Thesis

Electrical Engineering

Vaasa 2017



## BACHELOR'S THESIS

Author: Markus Nygård  
Degree Programme: Electrical engineering  
Specialization: Automation  
Supervisor: Erik Englund

Title: Implementation of APIs in Building Automation Systems

---

Date 25 April 2017

Number of pages 42

Appendices 1

---

### Abstract

This Bachelor's thesis was commissioned by ATN Automation in Vaasa, Finland. The thesis comprises instructions and a script program for retrieving and implementing weather data from APIs, supporting XML-documents, in building automation system. The script can easily be programmed to retrieve various weather data information from the past, the real-time weather and future forecasts. The software that is used for this is Schneider Electric's StruxureWare Building Operation Workstation.

The purpose with the thesis is to create a *Real-World Input* for implementing information from the Internet to building automation system. The main fields of information the company wanted me to research in was information concerning weather data and power consumption. The real-world information should then be available, e.g. in a table, for employees to use for programming.

The result is a script program that easily can be improved and further developed to retrieve more information from similar and other sources. The script was merged into SBO on the company's local server and a table of real-world inputs is available for use in building automation system programming.

---

Language: English

Key words: RWI, API, XML, Schneider Electric, ATN,  
Building Automation Systems

---

## EXAMENSARBETE

Författare: Markus Nygård  
Utbildning och ort: Elektroteknik, Vasa  
Inriktningsalternativ: Automationsteknik  
Handledare: Erik Englund

Titel: Implementering av API:er i fastighetsautomationssystem

---

Datum 25.04.2017

Sidantal 42

Bilagor 1

---

### Abstrakt

Detta examensarbete var gjort åt ATN Automation i Vasa, Finland. Examensarbetet omfattar instruktioner och ett skriptprogram för att hämta och implementera väderdata från API:er, som stöder XML-dokument, i fastighetsautomationssystem. Programmet kan lätt omprogrammeras till att hämta olika sorters information om väderdata som har varit, i realtid och framtida väderprognoser. Mjukvaran jag har använt mig av är StruxureWare Building Operation Workstation av Schneider Electric.

Syftet med detta examensarbete var att skapa en *Real-World Input* för implementering av information från Internet i fastighetsautomationssystem. De områden som företaget ville att jag skulle forska i var information beträffande väderdata, och elpriser och energiförbrukning. Real-World informationen skall sedan vara tillgängliga, t.ex. i en tabell, för de anställda att använda i sin programmering.

Resultatet är ett skriptprogram som enkelt kan användas, förbättras och vidareutvecklas för att hämta mera information från liknande eller andra källor. Programmet implementerades i SBO-mjukvaran i företagets lokala server och en tabell bestående av de i detta skede tillgängliga real-world data finns nu att användas i programmeringen för automationssystem.

---

Språk: engelska

Nyckelord: RWI, API, XML, Schneider Electric, ATN,  
Building Automation Systems

---

## OPINNÄYTETYÖ

Tekijä: Markus Nygård  
Koulutus ja paikkakunta: Sähkötekniikka, Vaasa  
Suuntautumisvaihtoehto: Automaatiotekniikka  
Ohjaaja: Erik Englund

Nimike: Taloautomaation API käyttöönotto

---

Päivämäärä 25.04.2017

Sivumäärä 42

Liitteet 1

---

### Tiivistelmä

Tämä opinnäytetyö on tehty ATN Automaatiolle, joka sijaitsee Vaasassa. Opinnäytetyö sisältää käyttöohjeet ja script-ohjelman, jolla on haettu ja toteutettu säätietoja API:stä, jotka tukevat XML-dokumentteja, kiinteistöautomaatiohjelmoinnissa. Ohjelma voidaan helposti ohjelmoida uudestaan saadakseen erilaisia tietoja: historiallisia säätietoja, reaaliaikaisia säätietoja sekä tulevaisuuden ennusteita. Ohjelmistona olen käyttänyt StruxureWare Building Operation Workstationia, joka on Schneider Electricin tekemä.

Opinnäytetyön tarkoituksena oli muodostaa *Real-World Input* Internet-tietojen toteutusta varten kiinteistöautomaatiojärjestelmässä. Säädata, sähköhinnat ja energiankulutus olivat tutkimuksen aiheena. Real-World-informaation pitää olla työntekijälle saatavilla ohjelmoinnissa, esim. taulukoiden muodossa.

Tulos on script-ohjelma, jota voi helposti käyttää, parantaa ja jatkuvasti kehittää, jotta saataisiin lisää informaatiota vastaavanlaisista lähteistä. Ohjelma toteutettiin SBO ohjelmistossa yrityksen paikallisilla servereillä. Lisäksi laadittiin taulukko, jossa historialliset, nykyaikaiset sekä ennustetut arvot näkyvät.

---

Kieli: englanti

Avainsanat: RWI, API, XML, Schneider Electric, ATN,  
Building Automation Systems

---

# Table of Contents

Table of Figures.....	vii
List of Appendices.....	viii
Abbreviations .....	ix
Preface .....	x
1 Introduction .....	1
1.1 Background.....	1
1.2 Purpose.....	1
1.3 Previous research and literature .....	2
1.4 Problems .....	2
1.5 Company.....	2
2 Intelligent Buildings and Building Automation .....	3
2.1 Energy efficiency.....	3
2.1.1 Benefits.....	3
2.2 Technologies.....	4
2.2.1 Process control.....	4
2.2.2 Air-conditioning systems.....	4
2.2.3 Central chilling systems .....	5
2.2.4 Lightning-control systems .....	5
2.2.5 Security and safety control systems .....	5
2.2.6 Communication and integration .....	5
2.3 Control strategies for buildings.....	6
2.4 Internet of Things.....	7
3 Building Automation Systems.....	8
3.1 Function .....	8
3.2 PLC .....	8
3.3 Monitoring .....	9
3.4 Area of application.....	9
4 Schneider Electric.....	10
4.1 Automation Server.....	10
4.2 StruxureWare Building Operation .....	11
5 Communication .....	13
5.1 TCP/IP .....	13
5.1.1 Features.....	13
5.1.2 TCP/IP Protocol Architecture.....	14
5.2 BACnet .....	15
5.3 API.....	16
5.3.1 API for commercial users .....	16

5.3.2	API for developers.....	16
5.3.3	History .....	16
5.3.4	API Types .....	17
5.3.5	Web Service.....	17
5.3.6	REST API.....	18
5.4	XML.....	19
5.4.1	DTD .....	21
6	Weather Services .....	23
6.1	Yr.no .....	23
6.2	Finnish Meteorological Institute .....	23
6.3	Foreca Ltd .....	24
6.4	INSPIRE .....	24
6.5	Open Geospatial Consortium.....	25
6.6	Web Feature Services .....	25
6.7	Comparison.....	26
7	Implementation of weather data in SBO .....	27
7.1	Retrieving data from API.....	27
7.2	Create a Simple XML Web Service interface.....	30
8	Result.....	35
8.1	Test.....	35
8.2	Graphics .....	36
9	Discussion.....	37
9.1	Challenges.....	37
9.2	Initial thoughts of the script program.....	38
9.3	Taking advantage of weather forecasts.....	38
9.4	Real-time energy information .....	38
9.5	Summary .....	39
10	Bibliography .....	40
	Appendices	

## Table of Figures

Figure 1.1 ATN logo .....	2
Figure 3.1 The control action of a PLC .....	9
Figure 4.1 Automation Server (Schneider Electric, 2017) .....	10
Figure 4.2 Script Editor .....	11
Figure 4.3 Graphics made in Graphics Editor .....	12
Figure 5.1 Layers in the TCP/IP protocol architecture.....	14
Figure 5.2 BACnet Architecture.....	15
Figure 5.3 Contents of the productlist.xml document .....	20
Figure 5.4 Contents of the productlist.dtd document .....	21
Figure 5.5 Contents of the productlistv2.xml document .....	22
Figure 7.1 Setting date and time.....	27
Figure 7.2 Creating date-time string.....	27
Figure 7.3 Creating a message string.....	28
Figure 7.4 Sending the web request.....	28
Figure 7.5 Part of the requested XML document from FMI .....	28
Figure 7.6 Calling the GetElementFMI function.....	29
Figure 7.7 Extracting a node.....	29
Figure 7.8 Inputs.....	29
Figure 7.9 Outputs .....	30
Figure 7.10 Screenshot of Yr.no web page .....	30
Figure 7.11 Creating a new interface.....	31
Figure 7.12 Creating a Simple XML Web Service .....	31
Figure 7.13 URL to Yr.no .....	32
Figure 7.14 Creating a new value .....	32
Figure 7.15 Creating an analog value .....	33
Figure 7.16 Selecting the XPath.....	33
Figure 7.17 List View .....	34
Figure 7.18 Information about a value .....	34
Figure 8.1 Function blocks in simulation mode .....	35
Figure 8.2 Trends from the test .....	36
Figure 8.3 Graphic of weather station (Template retrieved from Barbara Shea, edited by Markus Nygård).....	36

## List of Appendices

Appendix 1          Weather data comparison



## Abbreviations

API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ASHRAE	American Society of Heating, Refrigerating and Air Conditioning Engineers
BACnet	Building Automation and Control Network,
BAS	Building Automation System
BMS	Building Management System
B2B	Business-to-business
B2C	Business-to-consumer
CAD	Computer-Aided Design
CCTV	Close Circuit Television
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DTD	Document Type Definition
E-commerce	Electronic Commerce
FMI	Finnish Meteorological Institution
GML	Geography Markup Language
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
HVAC	Heating, Ventilation, and Air-Conditioning
INSPIRE	Infrastructure for Spatial Information in Europe
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
LAN	Local Area Network
MEP	Mechanical, Electrical and Plumbing system
NRK	Norsk Rikskringkasting
OGC	Open Geospatial Consortium
OSI	Open Systems Interconnection
PLC	Programmable Logic Controller
REST	Representational State Transfer
RWI	Real World Input
SBO	StruxureWare Building Operation
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WFS	Web Feature Service
WWW	World Wide Web
W3C	World Wide Web Consortium
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language

## **Preface**

I would like to thank Mats Nordgren and Petri Tikkala at ATN Automation for the opportunity to do my Bachelor's thesis. And I would also like to thank my supervisor Erik Englund at Novia University of Applied Sciences.

*Markus Nygård*

*Vaasa, April 2017*

# 1 Introduction

This Bachelor's thesis was made for ATN Automation and the work was carried out during the winter of 2016-2017. The thesis comprises instructions and a script program for retrieving and implementing weather data from APIs, supporting XML-documents, in building automation system. The script can easily be programmed to retrieve various weather data information from the past, the real-time weather and future forecasts. The software I will be using is Schneider Electric's StruxureWare Building Operation.

## 1.1 Background

The building automation industry is evolving due to new technologies and possibilities to connecting devices to the Internet. Electrical engineering companies and software developers are trying to keep up with the increasing demands, and they are implementing functions in their products that make it possible for users to connect to the web and use different web services. This makes it possible for us to benefit from the increasing amount of data and information provided by companies and organizations from around the world.

When talking about building automation there are traditionally four types of I/O points that are connected to the PLC system. These are the analog input (AI), analog output (AO), digital input (DI), and digital output (DO). Because of these possibilities to use real-time information from the Internet, ATN needs a way to manage these I/O points in their project planning and for this reason, they are imagining a fifth type of I/O point you could call a *Real-World Input (RWI)*. This I/O point could consist of e.g. implementation of weather data, energy prices in real-time, camera, or access control. Implementation means retrieving information from the Internet to affect parameters in e.g. thermal regulation in buildings.

## 1.2 Purpose

The purpose with the thesis is to create a *Real-World Input* for implementing information from the Internet to building automation system. By using information from the Internet, energy consumption and programming can benefit from using real-time and future values and data. The task given to me was to do a research of what kind of information is available. The main fields of information the company wanted me to research in was information

concerning weather and power consumption. The real-world information should then be available, e.g. in a table, for employees to use for programming.

A long-term goal with the thesis is to create a program that can easily be improved and further developed to retrieve information from other sources. Also, knowing and understanding of how to handle the information is an important factor. If real-world data is used correctly and successfully, it might even replace some hardware and other components.

### **1.3 Previous research and literature**

There is a long technical reference guide about the StruxureWare Building Operation software and there is only some brief information considering handling APIs. On Schneider Electric's exchange community one can find other users asking for more information about APIs, but at the time of this task there were no complementary literatures about it.

### **1.4 Problems**

Since there are several different types of APIs and none of the documented examples were useful for me, I had to learn about different APIs from other industries. This turned out to be quite challenging when trying to implement it into SBO. Another challenge was to learn a new programming language and the combination of these two problems turned out to be a lot more time consuming than expected.

### **1.5 Company**

Automation T&N Ab (ATN) was founded in 2007 and operates in the automation industry. Their work includes contracts in building automation systems, service and maintenance of these in everything from private homes to large commercial buildings and hospitals. ATN delivers easy to use automation solutions in heating and ventilation, and various control and monitoring systems for building services. They help their customers to save nature by reducing the energy consumption and providing the property's value through increased comfort and safety. The company has six employees. (ATN, 2016)



**Figure 1.1** ATN logo

## 2 Intelligent Buildings and Building Automation

The next generation of buildings provides new levels of comfort to the occupants with minimum possible energy consumption. The term for these buildings is *intelligent* or *smart buildings*. An integrated system is set to follow commands and rules but what makes a building smart is that it's proactively learn from occupant's behavior and adapt their operation based on the indoor and outdoor conditions. (Maasoumy & Sangiovanni-Vincentelli, 2016, p. 3).

### 2.1 Energy efficiency

Building control system plays a huge role when talking about preventing waste energy. The amount of energy required to run a building is determined by the *thermal efficiency*, e.g. thermal insulation and airtightness, *requirement of the indoor environment*, e.g. ventilation needs, lightning and hot water requirements, and *processes within the building*, e.g. IT equipment and industrial processes. These requirements taken together demand a level of base energy and provides a minimum level of energy expenditure. A reduction of base energy implies a change in building construction or use. Avoidable waste represents the difference between the base requirement and actual energy expenditure.

Some causes of avoidable waste include e.g. poor time and temperature control of the building, ineffective utilization of internal heat gains, plant oversizing, excessive ventilation, low operating efficiency of the HVAC system, poor system design and installation, standing losses, and unnecessary use of artificial lightning and air conditioning. A well-managed building has an avoidable waste level of below 15%. (CIBSE, 2009, pp. 1.4-1.5).

#### 2.1.1 Benefits

Different groups of user involved in a building will benefit differently from a modern control system. It is important to choose an appropriate type of control system because the benefits of one will only be obtained if the system is properly specified, installed, commissioned, operated and maintained. Different beneficiaries are e.g. the building owner, building tenant, occupants, facilities manager, and controls contractor. Some benefits for the building owner are higher rental value, the flexibility on change of building use, and individual tenant billing services. The tenant will on the other hand will benefit from reduced energy consumption, effective monitoring and targeting of energy consumption, and they will also have good control of internal comfort conditions. With the possibility of individual room control the

occupant will have better comfort due to lightning and effective response to HVAC-related complaints. A better monitoring system will help the facilities manager with supervision and rapid alarm indication and fault diagnosis. Bus systems will simplify installation for controls contractors and interoperability enlarges supplier choice. To summarize, an effective control system is essential to provide a productive, healthy and safe working environment for the occupants. The control system, along with good building design, plays a vital role in the prevention of energy waste and reducing the environmental impact of the building. (CIBSE, 2009, pp. 1.4-1.5).

## **2.2 Technologies**

A modern building contains of several technical services and some of the common services, more specific in the building automation industry, will be briefly introduced in this chapter.

### **2.2.1 Process control**

The basic tasks of building automation systems are installation management and control, where process control is one of the major functions. Most controlled processes in building systems are implemented with close-loop control. To achieve a stable control of the system there have been hundreds of control algorithms invented. Observations that have been made shows PID algorithms are used in over 80 per cent of closed control loops in industrial applications and this percentage may be even higher in building control systems. (Wang, 2015, p. 111).

### **2.2.2 Air-conditioning systems**

Thermal comfort is affected by several factors, e.g. air temperature, air velocity, radiant environment, clothing, relative humidity, and activity level. The most common measurement of comfort is by air temperature. To control parameters, such as temperature, air distribution, humidity, and indoor air quality, an air-conditioning system (also referred to HVAC system) is used. This is necessary in order to provide a healthy and comfortable indoor environment for people to work and live in. The basic controls of the air-conditioning system provide the control of thermal comfort and minimum health requirements, while the optimal control aims at providing satisfied thermal comfort and indoor air quality with minimum energy output. (Wang, 2015, p. 138).

### **2.2.3 Central chilling systems**

In most air-conditioned commercial buildings, central chilling systems consume about 25 to 50 per cent of annual energy budgets. The aim with a chilling system is to improve operation reliability and reduce overall energy consumption. This is achieved with a well-monitored and controlled system. Generally, the basic knowledge of a central chilling system is that it consists of three subsystems. These are *the chillers*, whose function is to produce an adequate quantity of chilled water at the required temperature, *the heat-rejection system*, which reject the heat from the refrigerant to the environment, and *the chilled water distribution system*, whose function is to distribute the chilled water to the user terminals. (Wang, 2015, p. 175).

### **2.2.4 Lightning-control systems**

In places for living, working, or other generic purposes where and when there is no sufficient daylight available, artificial lightning becomes essential for a visual environment. Lightning can be used for creating a dramatic or dynamic environment in special spaces, such as spaces for entertainment. Following the HVAC system in office and commercial buildings, the lightning system is one of the major energy consumers in buildings. The selection of lamps, the architectural layout and the control determine the energy efficiency of lightning systems and the quality of the visual environment provided. (Wang, 2015, p. 207).

### **2.2.5 Security and safety control systems**

Organizations, and individuals, use security and safety control systems to protect property, life, materials, and facilities against unauthorized entry, theft, fire, damage, and any other dishonest, illegal or criminal acts that might happen. Different systems have been made for different causes, e.g. close circuit television systems, access control systems, fire alarm systems, and burglar alarm systems. Security measures taken do not only comply with governmental regulations, but also for providing enhanced protection for employers, employees, customers, and other users of buildings. (Wang, 2015, p. 224).

### **2.2.6 Communication and integration**

Communication and integration are some of the main features of modern building automation systems and intelligent building systems. There might be many digital stations or devices to be integrated in a modern building, and local area networks are the primary choice for data transmission between these stations or devices within a short distance. To communicate between different devices in the BAS industry, different communication

standards have developed, such as BACnet, Lon Works, Modbus, and PROFIBUS. The Internet is a very popular tool and therefore Internet protocols, and mainly the most popular TCP/IP protocol, are used. The popularity of the protocols is due to they can be used to communicate across any sets of interconnected networks. (Wang, 2015, pp. 43-96).

### **2.3 Control strategies for buildings**

Generally, the control strategies for a building are the interaction between all the subsystems in a system and its different operating modes. For example, a complete HVAC system consists of an assembly of plant modules, many of which interact with each other. For every new building, there are inevitably combinations of subsystems that have not previously been documented and it is essential that a variety of possible, likely, and unlikely situations is drawn up and predicted responses of the control system followed through. Failure to consider these interactions may result in system instabilities, high energy consumption, and compromise of safe operation.

The HVAC plant can operate at different times by using operating modes. Possible modes could include:

- Normal operating during the occupied period, subdivided into: heating, cooling, natural or mechanical ventilation, thermal storage operation.
- Low outside air temperature interlock.
- Low return water temperature interlock.
- Plant shut-down.
- Fan overrun for air systems, pump overrun for water systems.
- Optimum start heating/cooling.
- Night cooling.
- Fire.

Depending on the nature of the HVAC system, the number of modes may vary. The operating modes should be listed systematically together with the operating states of the component subsystems, so that unsatisfactory situations will be easier to identify. Careful



attention is required to produce a combination of user and automatic control that is neither wasteful nor over-complex and self-defeating. (CIBSE, 2009, pp. 6.1-6.10).

## 2.4 Internet of Things

*“The Internet of Things (IoT) is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction”.* (Rouse, 2016).

Historically, building functions such as HVAC, security, and lightning systems have been operated as standalone entities. Hence, they have for the most part occupied a proprietary niche, separated from mainstream IT standards and systems. However, companies and resellers focusing on building automation are beginning to integrate a more cohesive environment among their customer’s systems. With the IoT concept, a range of devices and sensors are given IP addresses to communicate wirelessly over the Internet. Smart devices will become more intelligent and self-governing in how they share data with the building automation system and the cloud. Data analytics functions and administrative user interfaces will be provided by the cloud. Data crunching will help management to make informed decisions on how to reduce operating costs and energy consumption. This enables the building to optimize its role to suit the best purpose of its state at the time. (Moore, 2014).

There are several factors to encourage building managers and owners to take a fresh look at automation. One factor is that it saves administrative costs in the integration since it is more efficient to manage systems with one console as opposed to several. Also, concerns over obsolescence plays a role. (Moore, 2014).

But there are also reasons to be cautious about IoT. A survey done by Kalle Koutonen indicates that reasons companies doesn’t want to implement IoT could be that they are afraid for the safety or they have not adequate knowledge of the quite new concept. (Koutonen, 2015).

### 3 Building Automation Systems

Building Automation Systems (BAS), also known for Building Management Systems (BMS), are centralized, interlinked, networks of hardware and software, which monitor and control the environment in commercial, industrial, and institutional facilities. (KMC Controls, 2017).

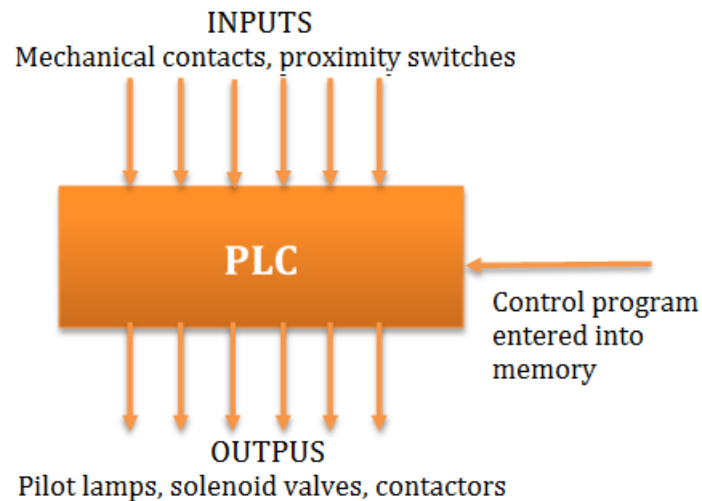
#### 3.1 Function

Generally, building automation begins with control of mechanical, electrical, and plumbing (MEP) systems. For instance, the heating, ventilation, and air-conditioning (HVAC) system is usually controlled, including control of its various pieces of equipment such as: chillers, boilers, air handling units, fan coil units, and heat pump units. Other systems that are often controlled and/or brought under a complete automation system include power monitoring, security, close circuit video (CCTV), card and keypad access, fire alarm system, elevators/escalators, and plumbing and water monitoring. Different systems are briefly described in Chapter 2.2. (KMC Controls, 2017).

#### 3.2 PLC

A Programmable Logic Controller (PLC) is a, as the name says, a programmable controller which is used in the automation industry. An apt definition of a PLC is that it is a “*digitally operating system, designed for use in an industrial environment which uses a programmable memory for internal storage of user-oriented instructions for implementing specific functions such as logic, sequencing, timing, counting and arithmetic, to control, through digital or analogue inputs and outputs, various types of machines or processes*”. (Crispin, 1997, p. 1).

The main building blocks a microprocessor-based system design are based on are the Central Processing Unit (CPU), memory, and input/output interface devices. The operation of a PLC is determined by a program written by the user. The program can be written in programming languages as e.g. ladder, function block programming or script. The memory saves the program and each successive instruction is fetched, decoded, and executed by the PLC. (Crispin, 1997, pp. 1-10).



**Figure 3.1 The control action of a PLC**

Control is achieved through in and outputs in the PLC system. The inputs are controlled with different types of sensors and control push buttons and switches. The outputs control the processes or the machines. The PLC system constantly checks the state of inputs and activates the respective outputs based on their instructions. (Crispin, 1997, pp. 1-10).

### **3.3 Monitoring**

Different kinds of sensors are used in building automation. The sensors help you to control and monitor the automation system and they also give you the opportunity to see graphic images of the building's measurements and provide an overview of the building's automated processes. There is also possible, for example the building's energy consumption and temperatures to be presented as diagrams from a certain time interval. (Forsell, 2013, p. 3).

### **3.4 Area of application**

Nowadays control systems are suitable for both small and large buildings. Control systems have been installed in larger buildings since the 1980s, and in the beginning of the 21<sup>th</sup> century automation have become increasingly common in private homes and cottages. (Forsell, 2013, p. 3).

## 4 Schneider Electric

Schneider Electric SE is a French global technology corporation that specializes in energy management and automation solution, spanning hardware, software, and services. Schneider Electric operates in over 100 countries with around 160 000 employees worldwide. (Schneider Electric, 2016).

### 4.1 Automation Server

An Automation Server is a hardware-based server that is factory programmed with StruxureWare Building Operation software. In a small installation, the embedded Automation Server acts as a stand-alone server, mounted with its I/O modules in a small footprint. In medium and large installations, functionality is distributed over multiple Automation Servers that communicate over TCP/IP.

Capable of coordinating traffic from above and below its location, the Automation Server can deliver data directly to you or to other servers throughout the site. The Automation Server can run multiple control programs, manage local I/O, alarms, and users, handle scheduling and logging, and communicate using a variety of protocols. Because of this, most parts of the system function autonomously and will continue to run as a whole, even if communication fails or individual servers or devices go offline.



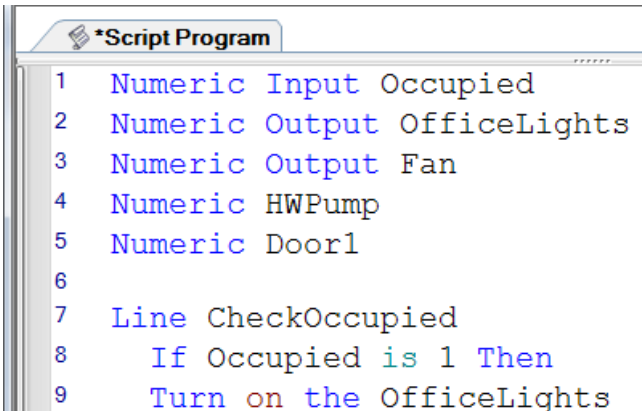
**Figure 4.1 Automation Server (Schneider Electric, 2017)**

One of the cornerstones of StruxureWare Building Operation is support for open standards. The Automation Server natively supports BACnet, LON, Modbus, web services, EcoStruxure Web Services and the I/O Bus. (Schneider Electric, 2017).

## 4.2 StruxureWare Building Operation

StruxureWare Building Operation (SBO) Workstation is a fully featured environment for operating and administering all aspects of the software. SBO is an interface that give both users and engineers access to their SmartStruxure solution server. The interface enables the users to monitor their energy usage and continuously improve the efficiency of their building. They can manage and view graphics, schedules, trend logs, alarms and reports. Users are required to have an account to access the Workstation, to ensure that individual IT policies for password formatting, aging, and uniqueness are followed. Engineers can maintain and configure all aspects of a SmartStruxure solution. By default, regional settings for units, time, and date formats are used in the software. It communicates using networking standards, such as DHCP, HTTP, and HTTPS, which makes it IT friendly and secure and makes installation easier. (Schneider Electric, 2014).

SmartStruxure have both Script and Function Block programming options, which ensures that the best programming method can be selected for the application. The script editor is based on an “everyday language statements” kind of programming language (Figure 4.2). Script supports Plain English controllers, so you don’t need to learn and program two different sets of formatting rules. With the Function Block editor, you create a program by adding function blocks and adding connections between them. Both script and function block programs or sections can be saved in a library so they can be inserted into other programs with ease. Data is easily exchanged between points, programs, servers, and other vendor’s systems via a simple binding mechanism. And the program runs even while it is being editing so there is no downtime. (Schneider Electric, 2014).



```
*Script Program
1 Numeric Input Occupied
2 Numeric Output OfficeLights
3 Numeric Output Fan
4 Numeric HWPump
5 Numeric Door1
6
7 Line CheckOccupied
8   If Occupied is 1 Then
9     Turn on the OfficeLights
```

Figure 4.2 Script Editor

You can create and edit graphics with the Graphics Editor (Figure 4.3). It is a tool for visualizing everything from the field control level to the enterprise level. There are a variety of tools to create whatever graphics required from a simple line to a photorealistic image. The Graphics Editor can import a wide variety of formats, including .jpg and CAD drawings. JavaScript can be used to customize behavior of the graphics, such as animation, which can highlight changes in the system or make navigation easier. You can, for example, with rotations and color codes indicate if a pump is running or not. (Schneider Electric, 2014).

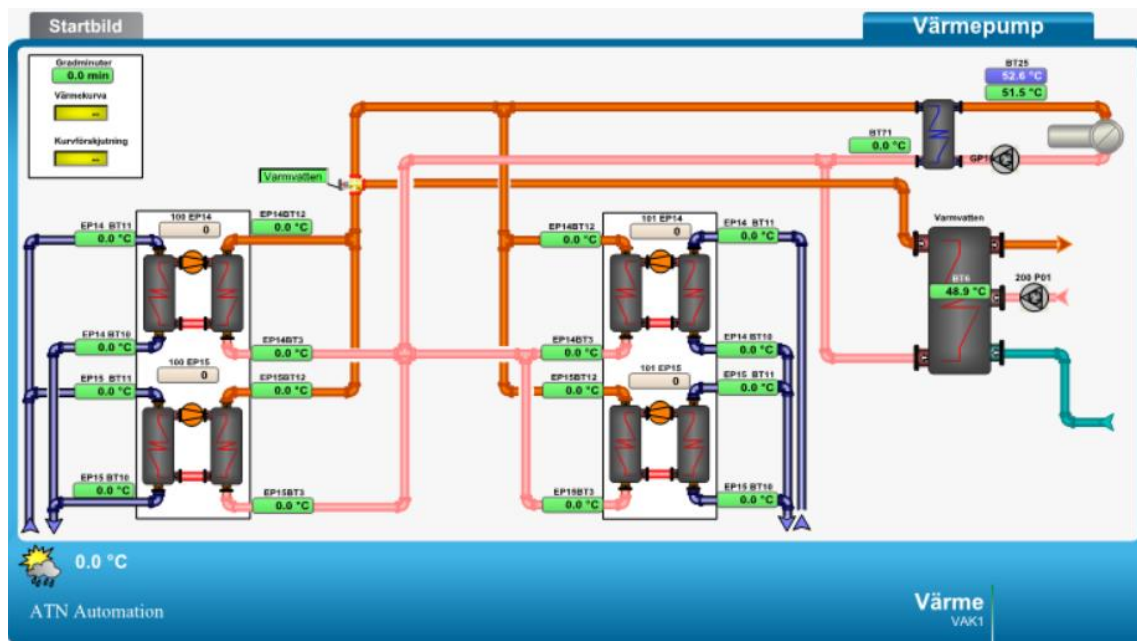


Figure 4.3 Graphics made in Graphics Editor

## 5 Communication

In all forms of networking, communication and understanding how the communication is used is an important factor. Therefore, various standards for data communication have been developed.

### 5.1 TCP/IP

TCP/IP was developed by Advanced Research Projects Agency (ARPA). In 1969 ARPA funded a research and development project to create an experimental packet-switching network called APRANET. It was built to study techniques for providing reliable and robust vendor-independent data communications, from where many modern data communication techniques were developed. The network became extremely successful and in 1975 it was converted from an experimental network to an operational network, and the responsibility for administering the network was given to the Defense Communications Agency. After the APRANET became operational, the development of the basic TCP/IP began. (Hunt, 1998, pp. 2-22).

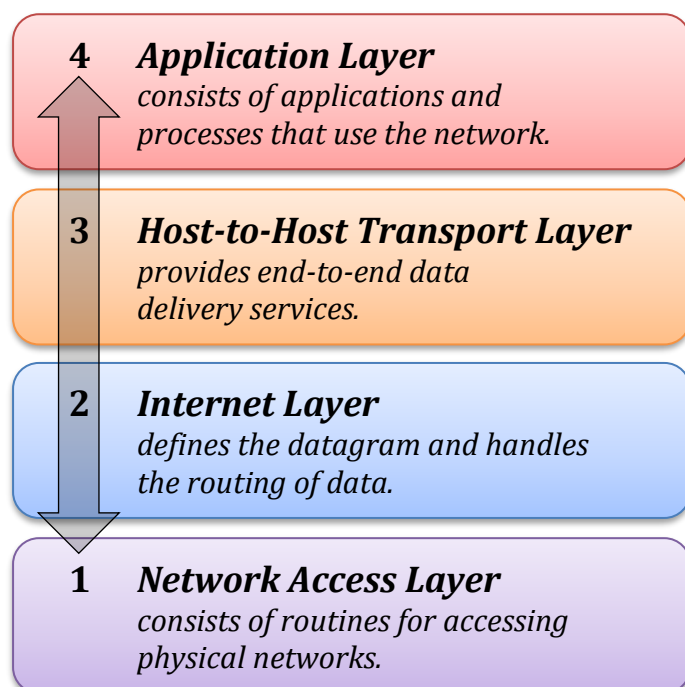
#### 5.1.1 Features

TCP/IP refers to an entire suite of data communications protocols and it got its name from two of the protocols that belong to it, *the Transmission Control Protocol* and *the Internet Protocol*. TCP/IP grew rapidly in population due to a few important features.

- It is open protocol standards that is available for free and is developed independently from any specific operating system or computer hardware. It is also ideal for uniting different hardware and software, even if you don't communicate over the Internet.
- It is developed independent from any specific physical network hardware, which allows TCP/IP to integrate several different kinds of networks.
- TCP/IP has a common addressing scheme. This means that it allows any TCP/IP device to uniquely address to any other device in the entire network.
- Its standardized high-level protocols for consistent and widely available user services. (Hunt, 1998, pp. 2-22).

### 5.1.2 TCP/IP Protocol Architecture

Generally, when describing TCP/IP with a layered model it is viewed as being composed of fewer layer than the seven layers used in the OSI model. Most descriptions define three to five levels in the protocol architecture. The model illustrated in Figure 5.1 is a four-level model which provides a reasonable pictorial representation of the layers in the TCP/IP protocol hierarchy. Data is passed down the stack when it is sent to the network, and when it's being received from the network data is passed up the stack. (Hunt, 1998, pp. 2-22).



**Figure 5.1** Layers in the TCP/IP protocol architecture

The **Application Layer** are application protocols as e.g. HTTP. Even administrative protocols are in this layer. The two most known is the DHCP and DNS protocols. **Internet Layer**, or the network layer, handles the logical addressing and the transmission and manipulation of data. Therefore, the Internet layers manages the IP protocol, but also other protocols. The **Transport Layer** ensures that the transmission of data works correctly between two parties over a network. Either this is done unreliable, which means that the protocol doesn't guarantee that information reach its destination. Or it is done in a reliably way and then the protocol checks that data arrives and if necessary transmits the data again. The most important protocols in this layer is the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) The **Network Access Layer** takes care of the contact between TCP/IP and the network. This layer is not always included since Ethernet in many cases manage the features and protocols that this layer contains. (Enqvist, 2016, pp. 12-15).



## 5.2 BACnet

BACnet, an acronym for Building Automation and Control Network, was developed under aegis of the American Society of Heating, Refrigerating and Air Conditioning Engineers (ASHRAE) to provide an open system specifically designed for the requirements of BMSs. It is now formally defined in *ANSI/ASHRAE standard 135-1995*. BACnet focused on defining a method for communications for the functions commonly found in building automation. (CIBSE, 2000, pp. 4.9-4.10)

BACnet consists of a collapsed architecture that corresponds to four layers of the Open Systems Interconnection (OSI) model. The seven-layer reference model is an international standard for conceptualizing network communication protocols and is widely used as the basis of many data communications standards.

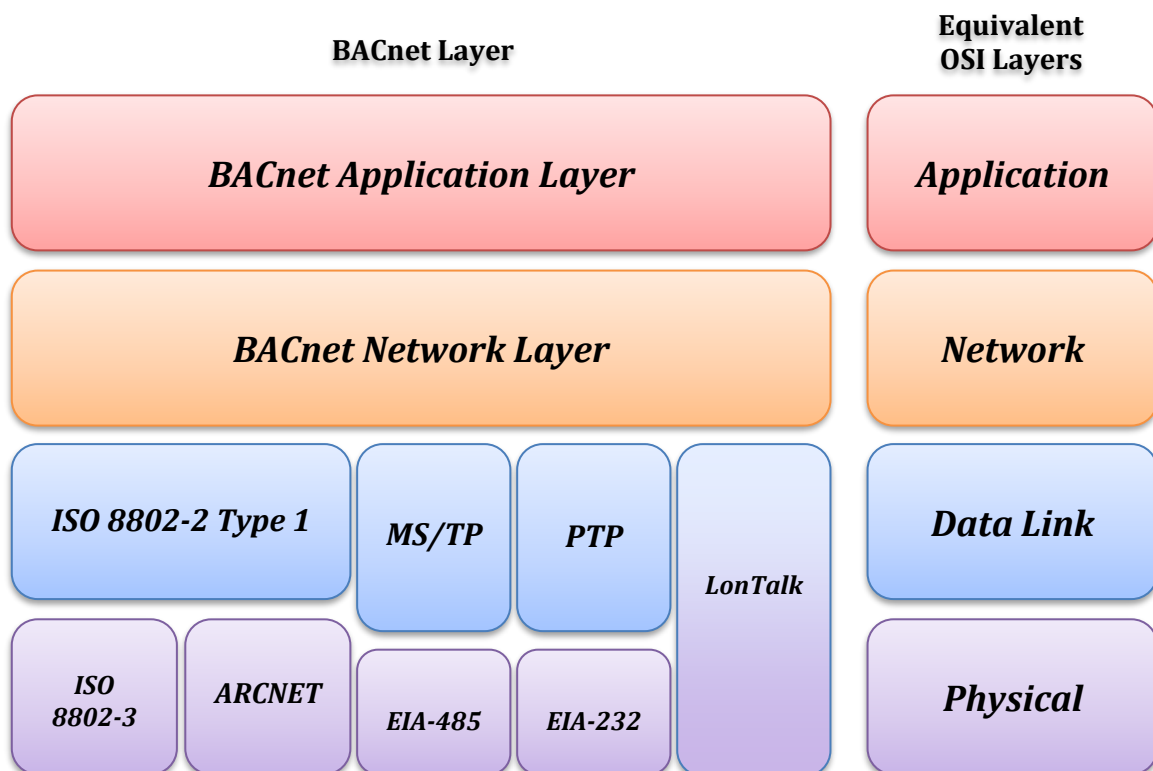


Figure 5.2 BACnet Architecture

The collapsed architecture was chosen due to the specific features and requirements of BAC networks, including a requirement that the protocol overhead be minimized as much as possible. BAC networks are local area networks (LAN) even though it is necessary in some applications to exchange information with devices in a building that is far away. Telephone networks are used for long-distance communication. The routing, relaying, and delivery issues are handled by the telephone system and can be considered external to the BAC network. (Cosby, 2007)

## 5.3 API

API is an acronym for *Application Programming Interface* and it is a set of commands, protocols, objects, and functions programmers can use to create software or interact with an external system. The API provides developers with standard commands for performing common operations so they don't have to write the code from scratch. (Techterms, 2016).

### 5.3.1 API for commercial users

Companies use API to share data in a controlled way. Most people will get in contact with APIs daily even if not realizing it, e.g. if you use a smartphone application to find out when the bus leaves or if you use another app to check the weather. There is (probably) an API that connects the app with the data from the local traffic department or the national weather service. (Trafiklab, n.d.).

### 5.3.2 API for developers

Developers can make requests by including calls in the code of their applications. The syntax is described in the documentation of the application being called. By providing a means for requesting program services, an API is said to grant access to or open an application. (Orenstein, 2000).

According to Josh Walker, an analyst at Forrester Research Inc. in Cambridge, Massachusetts, building an application without APIs *"is basically like building a house with no doors. The API for all computing purposes is how you open the blinds and the doors and exchange information"*. APIs also exist between applications. (Orenstein, 2000)..

### 5.3.3 History

The API concept was introduced in the late 90's, and as soon as the .COM bubble burst, platforms were looking for innovative ways to distribute data across e-commerce web sites. There was a need by software developers for standard ways to create a path between applications and allowing them to communicate with each other that lead to the development of APIs. Dissimilar from earlier existing software communication models, APIs came as a solution to this preoccupation because it utilized the ready proof HTTP protocol.

*“The concept of an API pre-dates even the advent of personal computing, let alone the Web, by a very long time! The principal of a well-documented set of publicly addressable “entry points” that allow an application to interact with another system has been an essential part of software development since the earliest days of utility data processing. However, the advent of distributed systems, and then the web itself, has seen the importance and utility of these same basic concept increase dramatically.” – Martin Bartlett (Lane, 2013).*

API originate from the software development during the Service Oriented Architecture (SOA) movement. The SOA movement found a more fertile environment in the world of start-up business, and therefore left the industry.

By launching their first distinct APIs in 2000, Salesforce and eBay became some of the pioneers in history of APIs. Other famous companies that followed the API development following years were e.g. Facebook, Twitter and Flickr. (Lane, 2013).

When an API is chosen, it is picked according to the service they provide and their reliability. One can define a stable API as one that maintains its services and is well documented. A good API will result in an increasingly number of clients, and an unstable or bad API will fade away since malfunctions and bugs in the application equals bad service, which will make the clients stop using it. (Bampende, 2015, pp. 2-6).

#### **5.3.4 API Types**

Web APIs are mostly available on two levels. A high level is accessible with specific languages such as Java or JavaScript. Low level uses direct HTTP requests to series of endpoints for accessing different parts of the application interface to their client. Regardless of the type of the web API, the core remains in the HTTP request. The client request will contain necessary information to be passed to the method invoked on the API for processing and information about the client browser. (Bampende, 2015, pp. 2-6).

#### **5.3.5 Web Service**

A Web Service is a computer program that identifies with a URI and whose interface and bindings can be defined, described, and discovered as XML objects. A Web Service supports direct interactions with other software agents using XML based messages exchanged via Internet-based protocols. (W3C, 2011) Web Services are not limited to the use of browsers

or hypertext markup language (HTML). Web technology such as HTTP is utilized for machine-to-machine communication and more precisely for transferring machine readable file formats such as XML and JSON. Those functions can be utilized to designate and retrieve the target information requested by the computer programs ran by the client. Web Service APIs are bound to provide good documentation to their clients, which is crucial for a better understanding of the services they dispense and its strength and its limitations. Failing to provide a good documentation will result in an incomprehensible API to use for developers. (Bampende, 2015, pp. 2-6).

### 5.3.6 REST API

The word REST stands for *Representational State Transfer*, and was first introduced by Roy Fielding in year 2000. REST is a web standard based architecture and it uses HTTP Protocol for data communication. It revolves around resources where every component is a resource that is accessed by a common interface using HTTP standard methods. In the architecture of REST, a REST Server provides access to resources and a REST client accesses and presents the resources where each resource is identified by global IDs or URIs. Various resources as Text, JSON, and XML can be represented by REST. (Tutorialspoint, 2017).

The HTTP methods that are the most commonly used in a REST based architecture are:

- GET – This provide a read-only access to a resource
- PUT – Used to create a new source
- DELETE – Remove a source
- POST – Update an existing resource or create a new resource
- OPTIONS – Fetch the supported operations on a resource

REST architecture based web services are known as RESTful Web Services. HTTP methods are used to implement the concept of REST architecture. A RESTful web service generally defines a URI, which is a service that provides resource representations as e.g. JSON or XML and a set of HTTP methods. (Tutorialspoint, 2017).

## 5.4 XML

XML is an acronym for *Extensible Markup Language* and it is a metalanguage that is used to represent and manipulate data elements. XML was created to simplify the exchange of structured documents, such as orders and invoices, over the Internet. The first XML 1.0 standard definition published by the World Wide Web Consortium (W3C) in 1998, set the stage for giving the real-world appeal to XML of being a true vendor-independent platform. XML has nowadays become somewhat a data exchange standard for e-commerce applications. (Coronel, et al., 2012, p. 500).

The Internet is used by companies to generate business transactions and these transactions are known as e-commerce (electronic commerce). E-commerce makes it possible for all kinds of organizations to sell products and services to a global market, where the transactions can take place between businesses (B2B) or between a business and a consumer (B2C). When buying a product online, the purchase order traveling over the Web used to be expected in the form of a HTML document. HTML tags describes how something *looks* on the webpage, e.g. heading styles and fonts, and often they come in pairs to start and end formatting features. The problem with this is that there is no easy way to extract details such as the order number, product code, quantity, or price from an HTML document. Since HTML documents only describes how to display the order in a Web browser, it doesn't permit the manipulation of the order's data elements. (Coronel, et al., 2012) To solve this problem, XML was developed. XML was designed for *describe* data and it was also made to be both human- and machine-readable. However, the XML tags are not predefined in the XML-file, you must define your own tags using a *DTD* to formally describe the data. (Refsnes, n.d.).

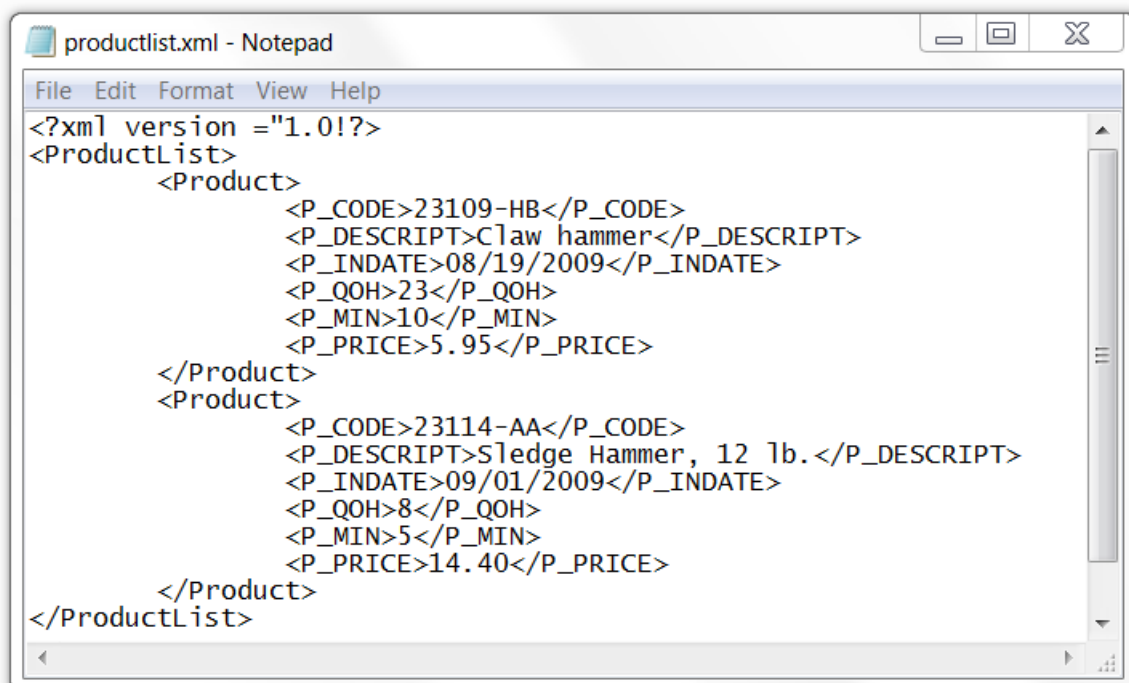
Some important characteristics:

- Definition of new tags to describe data elements is allowed.
- XML is case sensitive, e.g. <ProductID< is not the same as<Productid>.
- Tags must be properly formatted. All tags have both an open and a close tag, e.g. <ProductID>1234</ProductID>.
- XML must also be properly nested, e.g. a properly nested XML can look like <Product><ProductID>1234</ProductID></Product>.

- The *XML* and *xml* prefixes are reserved for XML only.

XML is not a replacement for HTML, but a complement. XML is used to describe and structure web data and HTML is used to format and display the same data. The new generation of HTML, *Extensible Hypertext Markup Language* (XHTML), is based on the XML framework. The idea with XHTML is to expand the HTML standard to also include features of XML, and though it is more powerful than HTML, it requires strict adherence to syntax requirements. (Coronel, et al., 2012, pp. 500-502).

To illustrate the use of XML for data exchange purposes, a Company A uses a XML document to send product data over the Internet to Company B. You can see the contents of the productlist.xml document in Figure 5.3.



```

productlist.xml - Notepad
File Edit Format View Help
<?xml version ="1.0!?">
<ProductList>
  <Product>
    <P_CODE>23109-HB</P_CODE>
    <P_DESCRIPT>Claw hammer</P_DESCRIPT>
    <P_INDATE>08/19/2009</P_INDATE>
    <P_QOH>23</P_QOH>
    <P_MIN>10</P_MIN>
    <P_PRICE>5.95</P_PRICE>
  </Product>
  <Product>
    <P_CODE>23114-AA</P_CODE>
    <P_DESCRIPT>Sledge Hammer, 12 lb.</P_DESCRIPT>
    <P_INDATE>09/01/2009</P_INDATE>
    <P_QOH>8</P_QOH>
    <P_MIN>5</P_MIN>
    <P_PRICE>14.40</P_PRICE>
  </Product>
</ProductList>

```

**Figure 5.3 Contents of the productlist.xml document**

The previous example illustrates a few important XML features:

- In the first line in the document you will find the XML document declaration represented, which also is mandatory for such file.
- On the following line, you will find a *root element*, which every XML document has. In this case, the ProductList is the root element.

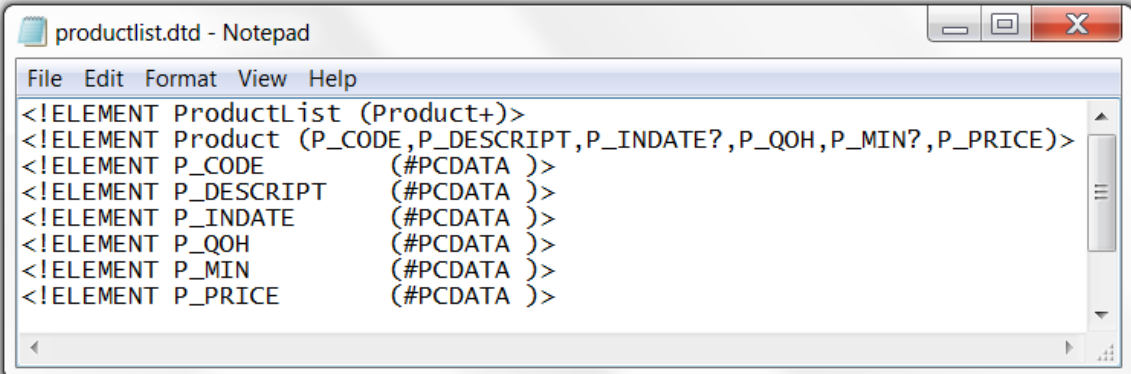
- The root element contains *child elements* or sub elements. On line 3, Product is declared as a child element of ProductList.
- For each element, there can be one or more sub elements. In this example, in every Product element there is several child elements, represented by P\_CODE, P\_DESCRIPT, P\_INDATE, P\_QOH, P\_MIN, and P\_PRICE.

The meaning of the XML is fairly self-evident, and Company B can process the document if it understands the tags made by Company A. A problem is that there is no certain way to validate the data or check whether the data are complete. You might encounter a P\_INDATE value of “09/01/2009”, but can you be sure it is correct? With *document type definition* one can address such concerns. (Coronel, et al., 2012, pp. 500-502).

#### 5.4.1 DTD

DTD, an acronym of Document Type Definition, is a file with a .dtd extension whose task is to give a description of the XML elements. This will just be a brief overview of the basics of DTD.

It is necessary to understand and validate each tag in a XML document, and a DTD file will provide the composition of the database’s logical model and defines valid elements or the syntax rules for each type of XML document. The following example, productlist.dtd in Figure 5.4, continues on the previous productlist.xml document in Figure 5.3.



```

<!ELEMENT ProductList (Product+)>
<!ELEMENT Product (P_CODE,P_DESCRIPT,P_INDATE?,P_QOH,P_MIN?,P_PRICE)>
<!ELEMENT P_CODE (#PCDATA )>
<!ELEMENT P_DESCRIPT (#PCDATA )>
<!ELEMENT P_INDATE (#PCDATA )>
<!ELEMENT P_QOH (#PCDATA )>
<!ELEMENT P_MIN (#PCDATA )>
<!ELEMENT P_PRICE (#PCDATA )>

```

**Figure 5.4 Contents of the productlist.dtd document**

The productlist.dtd file in Figure 5.4 provides definitions of the elements in the productlist.xml document, where one can note the following:

- The ProductList is declared as root element in the first line. The root element has one child, the Product element. The plus (+) sign tells you that Product appear one or more times within ProductList. (An asterisk (\*) would indicate that it occurs zero or more times).
- The second line describes the Product element and which its child elements are. A question mark (?) after P\_INDATE and P\_MIN indicates that they are optional elements.
- Line three to eight declares the child elements of the Product elements, where the #PCDATA keyword represents the actual text data.

After you have made a DTD file it must be referenced within the XML document. Figure 5.5 show the productlistv2.xml document where the productlist.dtd have been referenced in the second line.

```

productlistv2.xml - Notepad
File Edit Format View Help
<?xml version="1.0!?">
<!DOCTYPE ProductList SYSTEM "ProductList.dtd">
<ProductList>
  <Product>
    <P_CODE>23109-HB</P_CODE>
    <P_DESCRIPT>Claw hammer</P_DESCRIPT>
    <P_QOH>23</P_QOH>
    <P_PRICE>5.95</P_PRICE>
  </Product>
  <Product>
    <P_CODE>23114-AA</P_CODE>
    <P_DESCRIPT>Sledge Hammer, 12 lb.</P_DESCRIPT>
    <P_QOH>8</P_QOH>
    <P_MIN>5</P_MIN>
    <P_PRICE>14.40</P_PRICE>
  </Product>
</ProductList>

```

**Figure 5.5 Contents of the productlistv2.xml document**

Note that P\_INDATE and P\_MIN won't appear in all Product definitions, that's because they were declared as optional elements. A DTD file can be used by several XML documents of the same type, so there is no need to make a new one if e.g. product data is changed routinely in a XML document. (Coronel, et al., 2012, pp. 502-503).



## 6 Weather Services

One part of my thesis was to do a research in different weather forecasting and meteorological companies and institutes. I will compare what kind of weather data they offer, to decide which service I will use. There are both private owned companies and governmental agencies. All national institutes are required to offer free data, due to the *INSPIRE* directive (see Chapter 6.4), while private owned companies generally charge for their data or offer a very limited try out package.

### 6.1 Yr.no

Yr.no is a Norwegian website for weather forecasting and other meteorological information. The site is a collaboration, since 2007, between the Norwegian Broadcasting Corporation, *Norsk Rikskringkasting (NRK)*, and the Norwegian Meteorological Institute, *Meteorologisk Institutt*. The Meteorological Institute is responsible for the basic data and the technical content of Yr, and NRK for presentation, design, editorial content and delivery of notifications to users. On the website, you can find machine-generated weather forecasts (as graphs and maps) for 900 000 places in Norway and more than 9 million places around the world. (Yr.no, 2016).

Yr offers free weather data service and a lot of the information presented in yr.no are in XML-format and for this reason, parts of it (e.g. forecast for one location) may be used by other websites and online services. The free weather data service is very popular, with around 30 million downloads a day. (Nrk.no, 2014).

Yr have announced that they are working on a new and modern Yr-API that is predicted to be published in 2017. (Yr.no, 2016).

### 6.2 Finnish Meteorological Institute

The Finnish Meteorological Institute (FMI) is the organization in Finland responsible for gathering and reporting weather data and forecasts. The institute was founded in 1838 and got its current name 1969.

FMI offers an open data service that it possible for anyone to download and use its data. The data sets are available in digital format (machine-readable) (FMI, 2015). The ambition of the Open Data project was to make some of the institute's data freely available for public use

and the aim was to make it available for broader and more functional use, e.g. in the development of new applications, innovations, and services. The project came about as a result of the INSPIRE directive adopted in 2007, which requires governmental authorities to portray spatial data covered by the directive and to offer them for shared use via information networks. (Honkola, et al., 2013).

### **6.3 Foreca Ltd**

Foreca Ltd is a private Finnish weather forecasting company. It was first established in 1996 under the name *Weather Service Finland* and due to the prospect of expanding the company internationally in the beginning of 2001 the company name was changed to Foreca. (Lahdensivu, 2001).

Foreca offers an API that provides advanced weather data for locations worldwide based on Foreca's own weather forecasts. The coordinate system returns weather information for the closest location with up to ten days' forecasts. There is only a free trial valid for 1000 queries, and after that there is a fixed price starting at 2000€ monthly per server for the off-the-shelf configuration. Foreca Weather API supports JSON and XML. (Foreca, n.d.) Foreca also provides raw weather data available as XML and delimited ASCII. This service also includes a free, but very limited option for receiving weather data. (Foreca, n.d.).

### **6.4 INSPIRE**

INSPIRE stands for *Infrastructure for Spatial Information in Europe* and is an EU directive with conditions for the establishment of a geodata infrastructure in Europe. Geodata are descriptions of things that have a geographical location, such as constructions, lakes and roads, but also vegetation and population. Common synonyms of geodata are “geographic information” or “geographic data”. (Geodata.se, 2015).

The directive contains conditions for how we get access to geodata in a uniform way, and particular through search engines to see what data and services available in the EU, view services to watch and evaluate the content of a data set, and download services to access data to work with. The search and view services covered by the directive should be free of charge. (Geodata.se, 2015).

## 6.5 Open Geospatial Consortium

FMI's Open Data sets are distributed using open standard web service interfaces from geospatial data defined by the *Open Geospatial Consortium (OGC)* (FMI, 2014). OGC is an international voluntary consensus standards organization with members from governmental, commercial, academic and research organizations worldwide collaborate in a process encouraging development and implementation of open standards for geospatial content and services. OGC standards are technical documents which software developers use to build open interfaces and encodings into their products and services. When OGC standards are independently implemented in products or online services by two different software engineers, the resulting components should ideally work together without further debugging. OGC standards and supporting documents are available to the public at no cost. (OGC, 2017).

## 6.6 Web Feature Services

*Web Feature Services (WFS)* is a download service FMI use to provide a continuous online access to the most up-to-date data (FMI, 2014). The WFS is officially defined as “*The OGC Web Feature Service allows a client to retrieve and update geospatial data encoded in Geography Markup Language (GML) from multiple Web Feature Services.*” where the Geography Markup Language is the XML grammar defined to express geographical features. WFS refers to the sending and receiving of geospatial data through the medium of the World Wide Web (WWW), specifically HTTP. The two main methods of requesting data from a WFS server is using HTTP GET and HTTP POST. (Holmes & Pumphrey, 2008). The GET tag request data encoded into a URL and it takes form like this:

```
http://www.example.com/wfsserver?  
    name1=value1&  
    name2=value2&
```

This request sends the site “http://www.example.com/wfsserver” the key/value pairs of “name1=value1” and “name2=value2”. These values refer to a server-specific setting and as there is no official limit to the length of a URL, a fair amount of data can be passed through a GET request. But sending a large amount of data can be problematic due to it can become unwieldy and hard to read. (Holmes & Pumphrey, 2008).

FMI implements the simple WFS conformance class of the WFS 2.0 specification. The three main operations used are GetCapabilities, DescribeFeatureType, and GetFeature. Stored

queries are used extensively to select areas, features, and times, and parametrized queries are to be used with the GetFeature operation. Timestamp parameters are specified according in ISO 8601 format (e.g. 2017-12-24T00:00:00Z). (FMI, n.d.).

## **6.7 Comparison**

A research was made to compare weather data between Yr, Foreca, and FMI. Parameters requested were: temperature, precipitation, humidity, wind speed, wind direction, pressure, sunrise, sunset, and cloud. Also, the forecasts in minutes, hours, and days. The research was made in December in 2016 and is presented in the tables in appendix 1.

## 7 Implementation of weather data in SBO

There are two ways to implement weather data in SBO which I found best suited for this thesis. The first one I will present is by using APIs. To be able to retrieve data from an API I must make a script program using the Script Editor. This allows me to customize the program for any project that will use weather data. The weather data I will use for this method is retrieved from the Finnish Meteorological Institute APIs. The second method I will present use a built-in function called *Simple XML Web Service*. This method translates information in XML format into values, so it can be understood by Building Operation servers and displayed in WorkStation. Implementation of the web service is faster than using API, but not as versatile.

### 7.1 Retrieving data from API

One of the first things I must do in the code is to create a date and time string. FMI use time and date according to the ISO 8601 format. Date and time values are ordered in most-to-least significant ordered (year, month, day, hour, minute, second e.g. 2017-03-13T16:00:00Z). To create the string, I first save the date and time from the system to my own variables (Figure 7.1).

```
yyyy = Year
mo = Month
dd = DayOfMonth
hh = Hour
```

**Figure 7.1 Setting date and time**

As an example, yyyy = 2017, mo = 3, dd = 13, and hh = 16. Now I add lines, colons, letters T and Z, and fill out numbers lower than 10 with zeros (Figure 7.2).

```
If mo Is Less Than 10 And dd Is Less Than 10 And hh Is Less Than 10 Then
Starttime = yyyy;"-0";mo;"-0";dd;"T0";hh;";";"00:00Z"
Else
```

**Figure 7.2 Creating date-time string**

When the date and time string is done, the program proceeds to the creation of the message string which will be sent to the API. The core of the string consists of the *BaseURLStr*, *WFSservice*, and *MessageStr*. As seen in Figure 7.3, an API key is required to access data from FMI. Registration is needed to receive a key. The WFS service describe the service you are requesting. More info at [ilmatieteenlaitos.fi](http://ilmatieteenlaitos.fi). In the *MessageStr*, different queries,

such as location, time step, parameters and time-period, are inserted as strings and they are used together with the *GetFeature* operation. In this example, I request the current temperature and the temperature in six hours which is decided by input *enddt* (see Figure 7.8). They are sent as two different message strings, *MessageStr1* and *MessageStr2*.

```
APIkey = " "
BaseURLStr = "http://data.fmi.fi/fmi-apikey/";APIkey;"/wfs"
'Location = "Vaasa"
Parameters1 = "temperature" 'Temperature
Parameters2 = "temperature" 'Temperature Forecast, depending on endt
Timestep = "60"
WFSservice = "fmi::forecast::hirlam::surface::point::timevaluepair"
MessageStr1 = "?request=getFeature&storedquery_id=";WFSservice;"&place=";Location;\
               "&timestep=";Timestep;"&parameters=";Parameters1;"&starttime=";Starttime;\
               "&endtime=";Starttime;"
MessageStr2 = "?request=getFeature&storedquery_id=";WFSservice;"&place=";Location;\
               "&timestep=";Timestep;"&parameters=";Parameters2;"&starttime=";Endtime;\
               "&endtime=";Endtime;"
```

**Figure 7.3** Creating a message string

When the message string is ready, the function *SendWebRequest* sends the message to a server. The syntax for *SendWebRequest* is as follows:

```
SendWebRequest(BaseURL,Method,ContentType,Headers,Messages)
```

```
'*****
' Sending the web request to API
'*****
ResultXMLString1 = SendWebRequest (BaseUrlStr, "GET", "txt/xml", "", MessageStr1)
ResultXMLString2 = SendWebRequest (BaseUrlStr, "GET", "txt/xml", "", MessageStr2)
```

**Figure 7.4** Sending the web request

The web request is saved as a long XML document in *ResultXMLString1* and *ResultXMLString2*. In Figure 7.5 is the part of the first XML document we are interested in.

```
<wml2:point>
  <wml2:MeasurementTVP>
    <wml2:time>2017-03-13T16:00:00Z</wml2:time>
    <wml2:value>3.07</wml2:value>
  </wml2:MeasurementTVP>
</wml2:point>
```

**Figure 7.5** Part of the requested XML document from FMI

To extract the value from the XML document, the document is sent to a script function program called *GetElementFMI*. The function is called with the *ResultXMLString* and the keyword “*wml2:value*” attributes (Figure 7.6).

```

'*****
' Function "GetElementFMI" to parse the XML document and collect data
'*****
' Extract information from result
Temperature = GetElementFMI(ResultXMLString1, "wml2:value")
TemperatureForecast = GetElementFMI(ResultXMLString2, "wml2:value")

```

**Figure 7.6 Calling the *GetElementFMI* function**

As could be seen in Figure 7.5, the keyword *wml2:value* occurs two times and they work as open and close tags for the value. The script function locates these keywords and extracts the element between the tags. The element is returned and represented as an output in the script program.

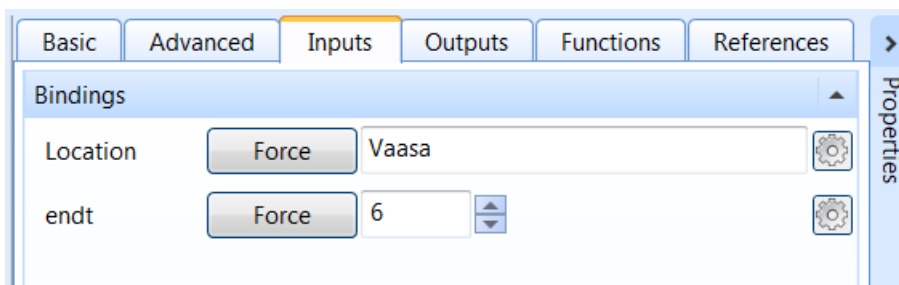
```

' Extract location
strStartMarker = "<";strElementName;>"
strEndMarker = "</";strElementName;>"
' Find the node
nStartIndex = Search(strText, strStartMarker) + length(strStartMarker) - 1
tempStr = right(strText, length(strText) - nStartIndex)
nEndIndex = Search(tempStr, strEndMarker) - 1
' Return the value
Return Mid(tempStr, 1, nEndIndex)

```

**Figure 7.7 Extracting a node**

In the example program it is possible to change location and the variable *endt*. *Endt* decides how many hours ahead the current time the forecast predictions should be. The **Inputs** can be changed from the WorkStation on the properties in the main program (Figure 7.8).



**Figure 7.8 Inputs**

Under the **Outputs** tab you can follow the output values in real-time as the XML document updates automatically (Figure 7.9).

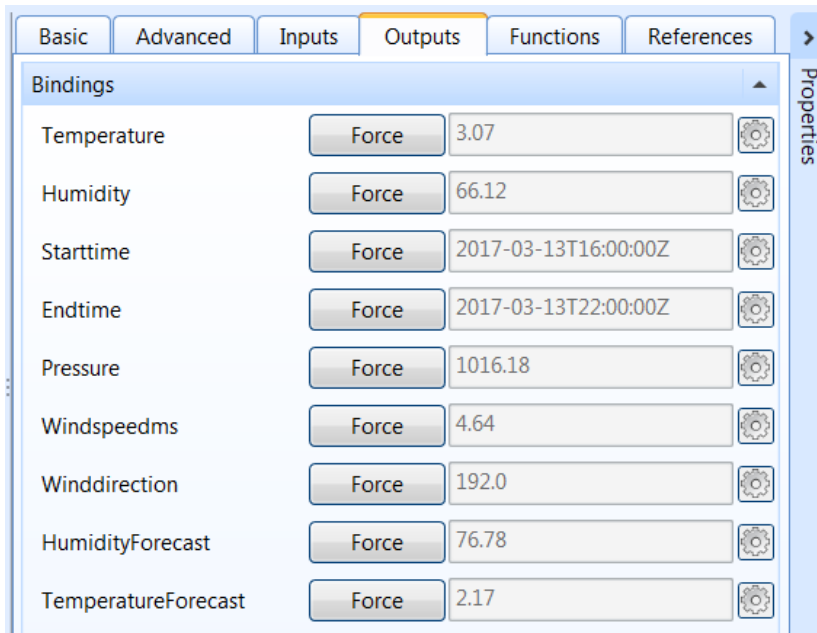


Figure 7.9 Outputs

## 7.2 Create a Simple XML Web Service interface

SBO has a built-in web service interface, called *Simple XML Web Service*, which is used for REST and standard XML files. This function will be used for retrieving weather data from Yr.no.

When creating the interface, the URL of a specific location from Yr.no will be used. In Figure 7.10 the web page has been navigated to the city Vaasa in Finland.

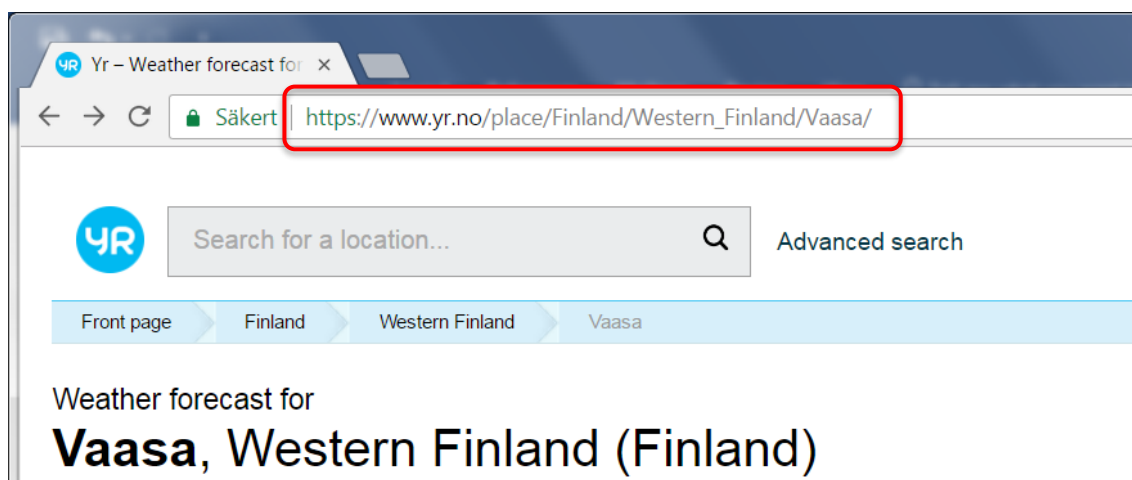
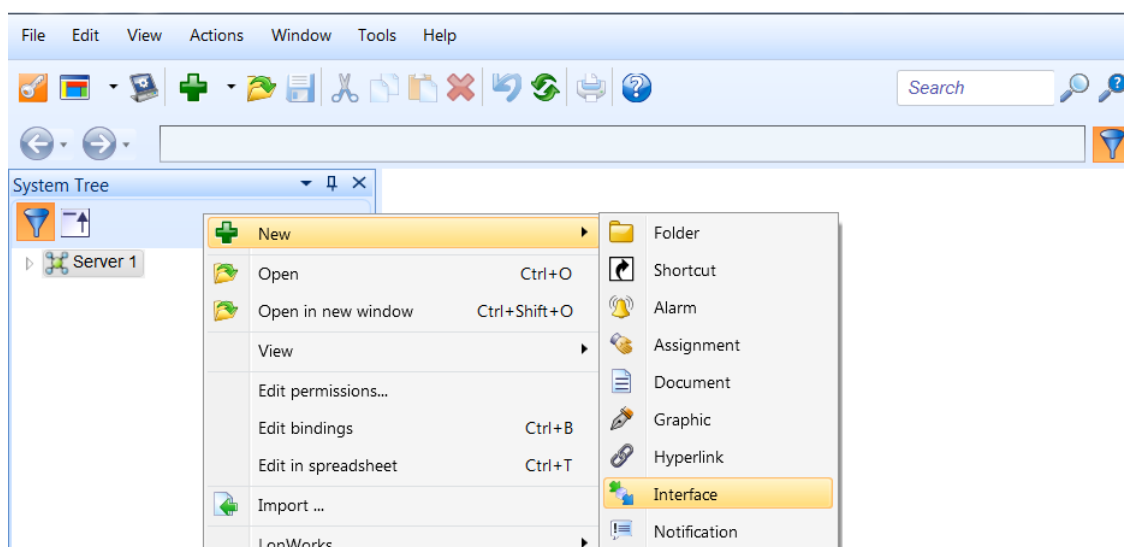


Figure 7.10 Screenshot of Yr.no web page

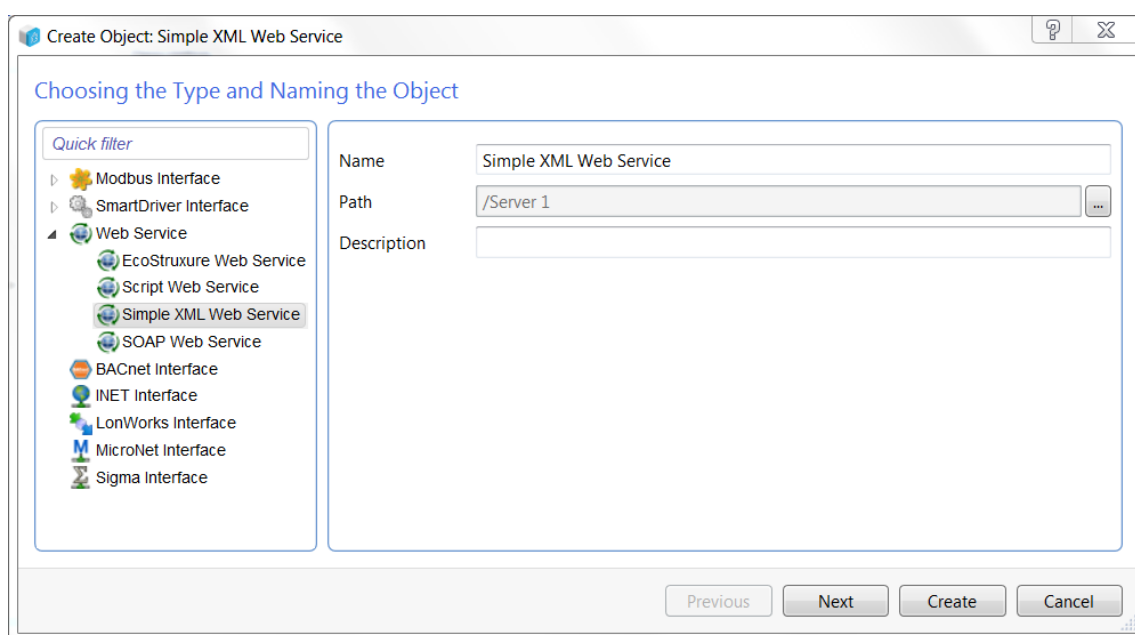


In WorkStation, select where in the **System Tree** you want to create the interface. It can be the server or a folder. Right-click where you want to want to create it and select **New** and **Interface**. (Figure 7.11).



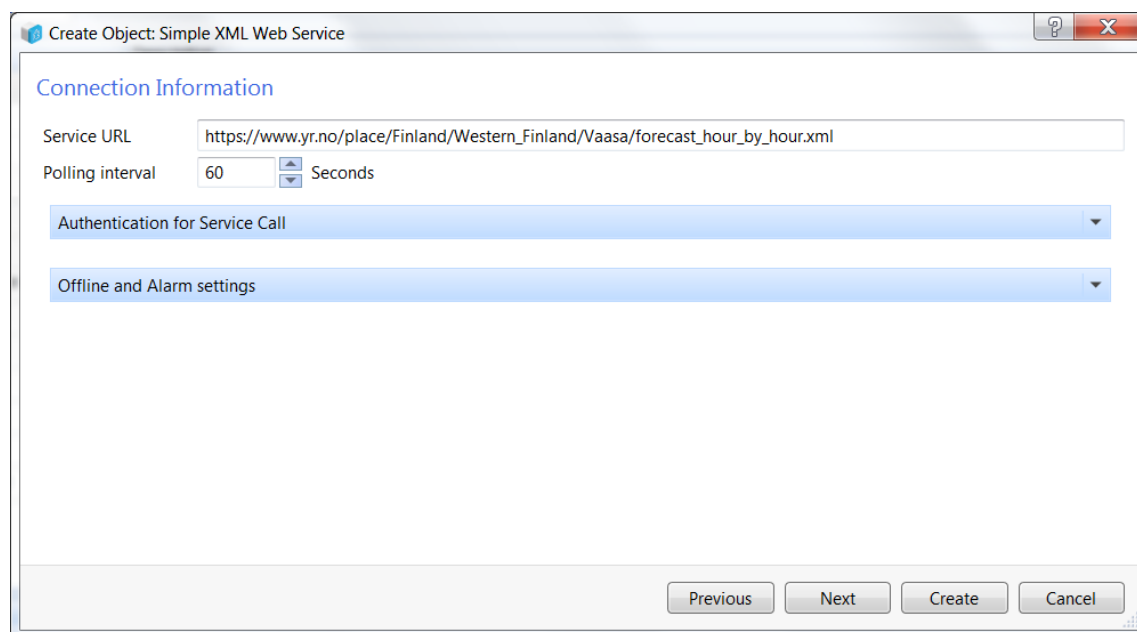
**Figure 7.11** Creating a new interface

Expand the **Web Service** tab and select **Simple XML Web Service**. Type a name for the interface in the **Name** box and a description in the **Description** box (optional). Then click **Next**. (Figure 7.12).



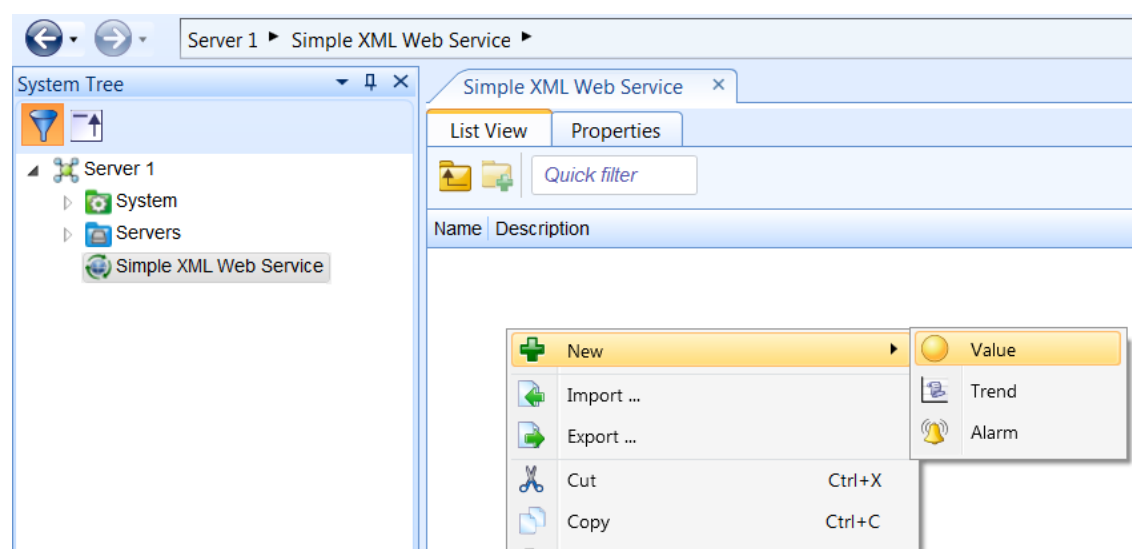
**Figure 7.12** Creating a Simple XML Web Service

Type or copy-paste the URL from Yr.no into the **Service URL** box. After the URL string, you can add either *vassel\_nu.xml*, *forecast\_hour\_by\_hour.xml*, or *forecast.xml*. For more information about the additions, check appendix 1. Then click **Create**. (Figure 7.13).



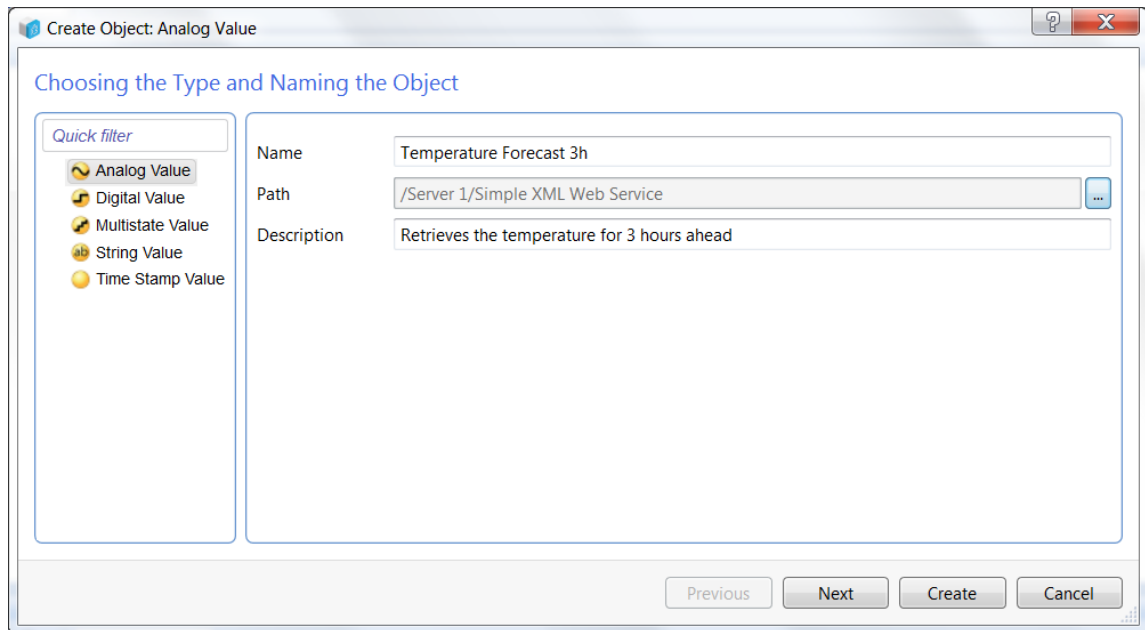
**Figure 7.13** URL to Yr.no

To add a value to the interface, right-click the **Simple XML Web Service** in the **System Tree** or on the work space in **List View**. Point to **New** and select **Value**. (Figure 7.14).



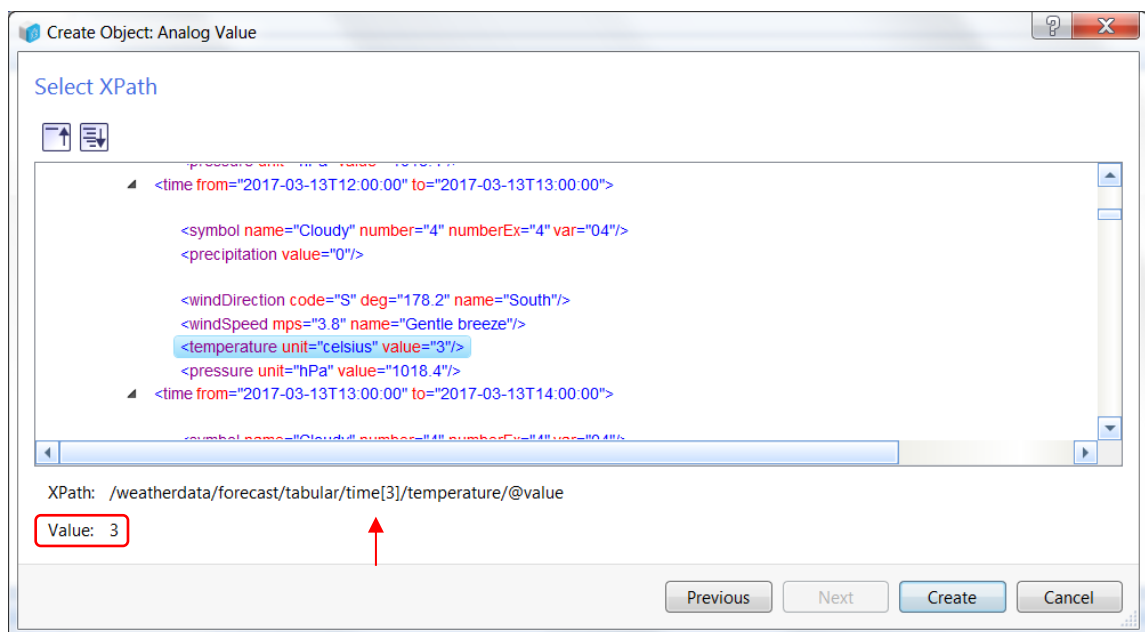
**Figure 7.14** Creating a new value

Choose a value from the object list and type a name and description for the value in the **Name** box, respectively in the **Description** box. Click **Next**. (Figure 7.15).



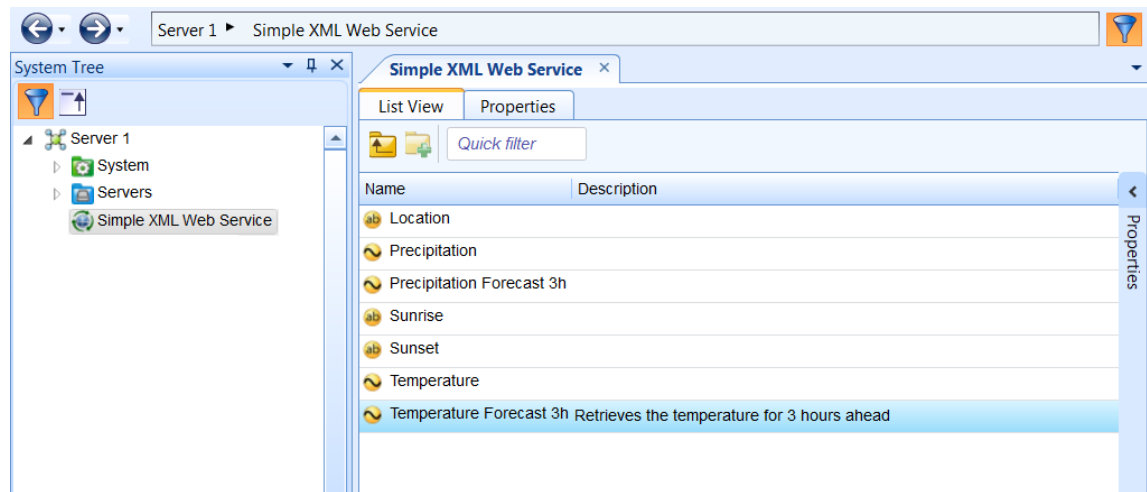
**Figure 7.15** Creating an analog value

In the Select XPath list, select the XPath to the value you want to add to SBO by clicking on the value. “time[3]” in the XPath string indicates that it is in the third tabular, which in this case means that it is the forecast for three hours ahead. If the value is correct, click **Create**. (Figure 7.16).



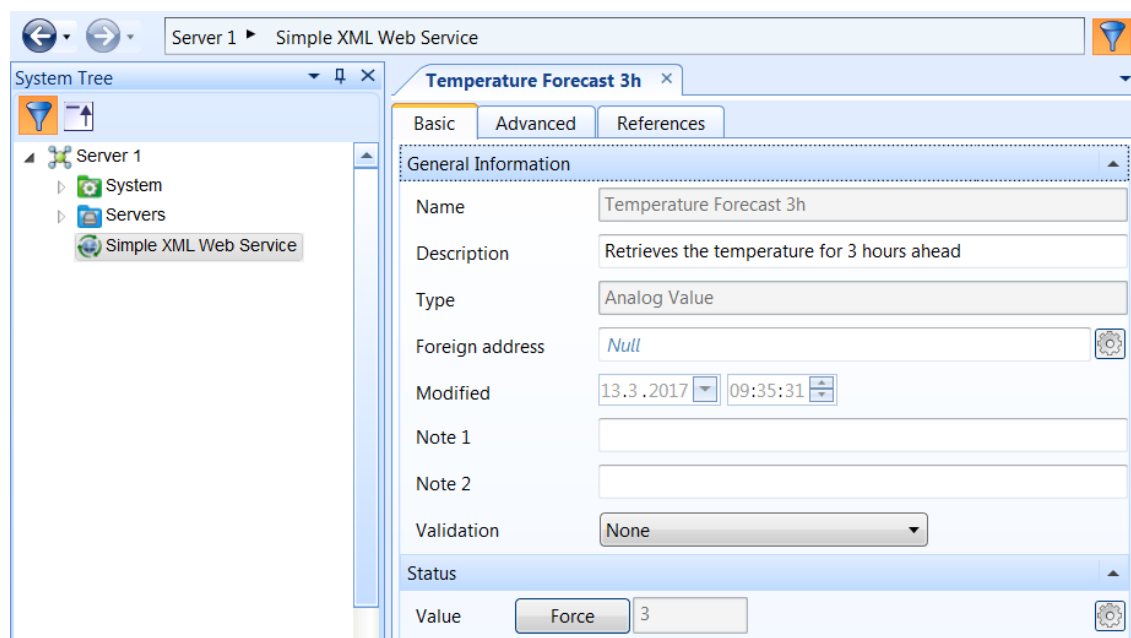
**Figure 7.16** Selecting the XPath

To see more information about a value, double-click the value in the **List View**. (Figure 7.17).



**Figure 7.17 List View**

It is possible to change the description below **General Information**. You can check the value below **Status**. If needed, the XPath can be changed by clicking the **Advanced** tab. (Figure 7.18).



**Figure 7.18 Information about a value**

## 8 Result

The goal for this thesis was to create a complementary input to the already existing I/O points, an input that we can call *Real-World Input*. A research was conducted to figure out what kind of online information is available from different organizations, that could be used in the field of building automation systems. The main areas researched in was information concerning weather data and energy consumption.

What I finally came up with is a script program that works as a template for future projects. The template is meant to work as a base, from where the user (programmer) can start building his own program for a specific project. The most important functions in the script program are described in Chapter 7.1. The program collects weather data from the Finnish Meteorological Institute and it is possible to change location, parameters, and time and date.

### 8.1 Test

One way to take advantage of knowing the weather forecast is to affect the heating system in a building ahead of the real-time measurements. For example, this could be done by comparing the current temperature to the forecast in e.g. six hours. If the forecast indicates a warmer temperature, the heating system could begin reducing the temperature in the building beforehand. In Figure 8.1 you can see a simple function block program that converts the difference between the current and the predicted temperature to a value that can be used to reduce the heating in a building.

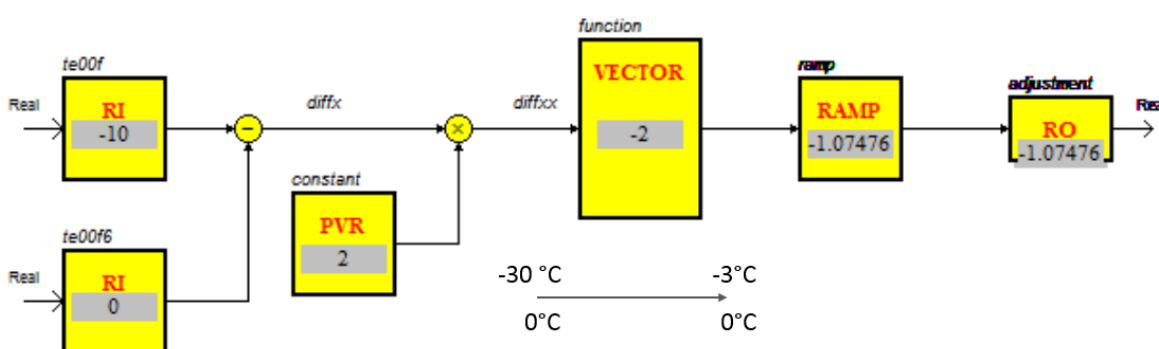
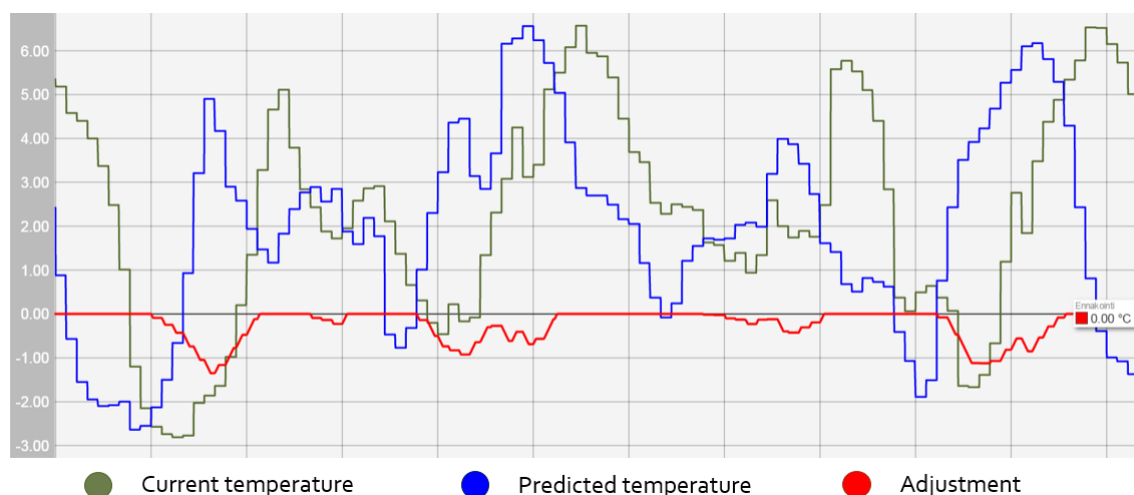


Figure 8.1 Function blocks in simulation mode

The predicted temperature (te00f6) is 10° warmer than the current temperature (te00f) and therefore the vector block will reduce the setpoint by 2°. A test was carried out on a real test object and trends from four days can be seen in Figure 8.2 on the next page.

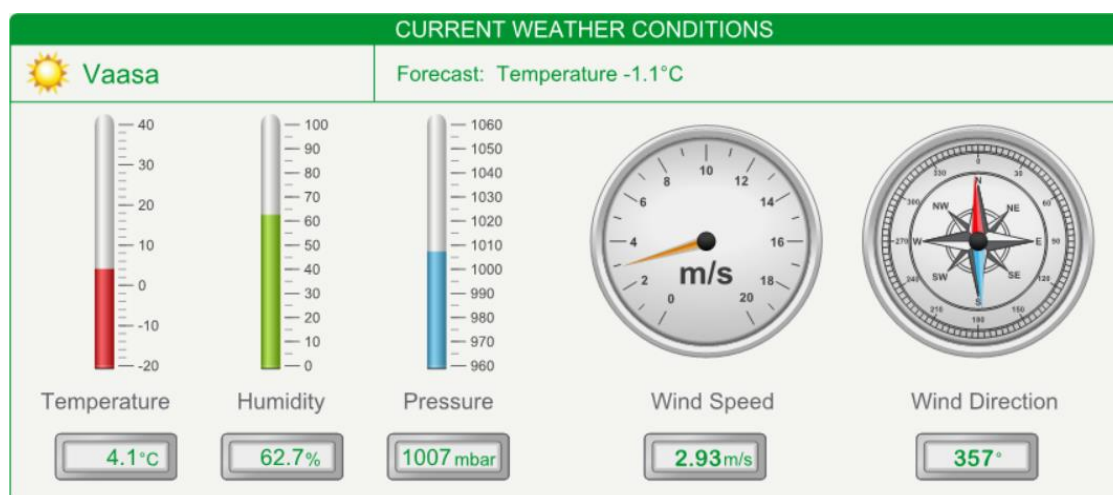


**Figure 8.2 Trends from the test**

The temperatures are sampled every hour and as can be seen in the graph, when the predicted temperature is higher than the current, an adjustment is made. Observe that this program is not a complete product, it only acts as an example on how one can benefit from the weather forecast.

## 8.2 Graphics

Graphics are used for interfaces for both the end-user and the developer. To make them more informative, weather data can be connected and animated on the graphic. In Figure 8.3 you can see a weather station with the current weather conditions and the temperature forecast.



**Figure 8.3 Graphic of weather station (Template retrieved from Barbara Shea, edited by Markus Nygård)**

## 9 Discussion

This thesis is part of my education and a requirement for graduation. I have learned to work independently on a larger project and face the unexpected problems that may occur. I have also learned how I better can distribute my time spent on different tasks.

I personally think that I ended up with a good result for this thesis and I believe ATN now has a good ground to start from for further develop the usage of online information. I got new experiences by getting acquainted with new programming languages and software. I got a lot of help from Schneider Electric's exchange community forum for learning programming languages from other users' programs and solutions. I learned that it was best to "keep it simple" when programming a script program, and the function block program is not hard per se when you once learn about all the different blocks.

### 9.1 Challenges

The most challenging and time consuming part of my thesis was to figure out how I should be able to access the online information, in which format it should be in, and how I could implement it. This was a new topic for me and it turned out to become a huge bottle neck. This also affected my motivation since, what it seemed like at the time, I didn't progress at all. I learned more about APIs over time and because of all the concerns with implementing it to the automation system, I have gotten an insight in how I can use and implement other APIs in the future, perhaps also to other systems.

Since I pretty much was given free rein to form the project, I could do it the way I wanted but mostly in a way necessary. This meant that I didn't get much tutoring, except for a crash course of the software. I spent a lot of time figuring out how I could implement online data to building automation systems, since there barely wasn't any information about it. Because it was time consuming I didn't have enough time to create a properly program, and therefore had to settle with a program that works as a template for future projects.

The theory part of this thesis is more about describing and giving a general aspect to the subjects. One reason for this is the lack of documentation about using APIs in building automation systems. The reader might not know what API or BAS is and therefore it seems more logical to describe them as two larger subjects rather than focusing too much on details.

## **9.2 Initial thoughts of the script program**

The result differs a bit from the expectations set in the beginning. But it didn't come as a surprise since no one exactly knew how it would turn out. When I finally figured out how I could implement the API to the software I didn't have enough time to create the program I initially thought of. This program would enable the user (programmer) to implement the program in any project and, e.g. with some buttons and menus, easily choose which parameters that should be used in the specific project. With some more experience with the software, this wouldn't be too difficult to do.

## **9.3 Taking advantage of weather forecasts**

This thesis was all about weather data and forecasts and therefore, there have been discussions on how one can use weather forecasts to its fullest. What was discussed with ATN is what I demonstrated in Chapter 8 about reducing the temperature of the indoor heating system when the weather forecast indicates warmer climate.

Other interesting discussions was held with my supervisor. We talked about the effects and benefits of usage of longer forecasts that could reach to several days or even weeks. For example, if you can predict when electricity prices will be low, the distributed electricity can be used while the buildings self-produced electricity can be stored. By predicting when the electricity prices will rise, the facility or building can avoid the high prices by using its stored energy or running only on its own system e.g. solar energy, thermal heating etc.

## **9.4 Real-time energy information**

One part of this thesis could have been about real-time energy prices and energy consumption. After being in contact with the local energy distributors, I got the perception that there were not any services corresponding to the weather APIs. Information they could provide with APIs are e.g. real-time electricity prices and energy consumption for statistics. I believe many parties would benefit from having an API that tells you e.g. real-time electricity prices and energy consumption. The occupants, building owner, controls contractor, and the energy distributor could all benefit from it. The contractor would offer the service to their client with the motivation of a more intelligent and more efficient automation system. The building owner might consider an electric company that can provide this service, therefore the energy distributor will be a more attractive option. If a building is



more efficient than it is possible that the energy cost will be lower and the environment will be better for the occupants.

## **9.5 Summary**

The whole idea by using real-time information is for better control and higher efficiency. Connecting devices to the Internet and using more online data will lead to a lesser need of hardware. The goal is always to strive towards a greener environment and one way to accomplish that is with the Internet. But there are always two sides of everything and new problems and challenges will occur.

This thesis has been a good introduction to the subject about BAS and API for me, and I believe that there is a lot of work that can be done further on. I will continue working with and coming up with new ideas to use weather APIs in building control systems. Later work can comprise investigations of other industries and the possibility to use their APIs.

## 10 Bibliography

ATN, 2016. *Automation T&N Ab*. [Online]

Available at: <http://www.atn.fi/>

[Accessed 8 February 2017].

Bampende, H. H., 2015. *Third Party Web API Implementation in Web Application Development*. Helsinki: Bachelor of Engineering, Metropolia University of Applied Sciences, Media Engineering.

CIBSE, 2000. *Building control systems*. London: Butterworth-Heinemann, pp. 4.9-4.10.

CIBSE, 2009. *Building Control Systems*. Norwich: Page Bros, pp. 1.4-1.5, 6.1-6.10.

Coronel, C., Morris, S. & Rob, P., 2012. *Database Principles: Fundamentals of Design, Implementation, and Management*. 10th International ed. Mason, Ohio: Course Technology/Cengage Learning, pp. 500-503.

Cosby, S., 2007. *BACnet Architecture*. [Online]

Available at: <http://www.chipkin.com/bacnet-architecture/>

[Accessed 9 March 2017].

Crispin, A., 1997. *Programmable logic controllers and their engineering applications*. 2nd ed. Maidenhead: McGraw-Hill Publishing Company, pp. 1-10.

Enqvist, M., 2016. *IEC 61850-konfiguration av REF 630 för DEMVE-projektet*. Vaasa: Examensarbete för ingenjörsexamen. Yrkeshögskolan Novia, Utbildningsprogrammet för elektroteknik.

FMI, 2014. *Open data manual*. [Online]

Available at: <http://en.ilmatieteenlaitos.fi/open-data-manual>

[Accessed 7 March 2017].

FMI, 2015. *The Finnish Meteorological Institute's open data*. [Online]

Available at: <https://en.ilmatieteenlaitos.fi/open-data>

[Accessed 13 February 2017].

FMI, n.d. *FMI Open data WFS services*. [Online]

Available at: <http://en.ilmatieteenlaitos.fi/open-data-manual-fmi-wfs-services>

[Accessed 7 March 2017].

Foreca, n.d. *BatchXML*. [Online]

Available at: <http://corporate.foreca.com/en/products-services/data/batchxml>

[Accessed 9 February 2017].

Foreca, n.d. *Weather API*. [Online]

Available at: <http://corporate.foreca.com/en/products-services/data/weather-api>

[Accessed 9 February 2017].

Forsell, J., 2013. *Byggnadsautomation ur en energieffektiv synvinkel*. Ekenäs:

Examensarbete för ingenjör (YH)-examen. Yrkeshögskolan Novia,

Utbildningsprogrammet för Automationsteknik och IT.

Geodata.se, 2015. *Inspire*. [Online]

Available at: <https://www.geodata.se/Vad/Inspire/>

[Accessed 7 March 2017].

- Holmes, C. & Pumphrey, M., 2008. *Using a Web Feature Service*. [Online]  
Available at: <http://www.ogcnetwork.net/wfstutorial>  
[Accessed 7 March 2017].
- Honkola, M.-L. et al., 2013. *The Finnish Meteorological Institute - Final report for the open data project*, Helsinki: Unigrafia.
- Hunt, C., 1998. *TCP/IP Network Administration*. 2nd ed. Sebastopol: O'Reilly & Associates, pp. 2-22.
- KMC Controls, 2017. *Understanding Building Automation and Control Systems*. [Online]  
Available at:  
[https://web.archive.org/web/20130519124213/http://www.kmcccontrols.com:80/products/Understanding\\_Building\\_Automation\\_and\\_Control\\_Systems.aspx](https://web.archive.org/web/20130519124213/http://www.kmcccontrols.com:80/products/Understanding_Building_Automation_and_Control_Systems.aspx)  
[Accessed 8 February 2017].
- Koutonen, K., 2015. *Nya internetbaserade möjligheter inom*. Raseborg: Examensarbete för ingenjör (YH)-examen. Yrkeshögskolan Novia, Utbildningsprogrammet för Automationsteknik och IT.
- Lahdensivu, M., 2001. *Weahter Service Finlandista tuli*. [Online]  
Available at: <http://www.is.fi/digitoday/art-2000001333945.html?nomobile=4>  
[Accessed 9 February 2017].
- Lane, K., 2013. *API Evangelist: History of APIs*. [Online]  
Available at: [history.apivangelist.com](http://history.apivangelist.com)  
[Accessed 14 February 2017].
- LVM, 2013. *Finnish Meteorological Institute makes its data sets public*. [Online]  
Available at: <https://www.lvm.fi/-/finnish-meteorological-institute-makes-its-data-sets-public-738767>
- Maasoumy, M. & Sangiovanni-Vincentelli, A., 2016. *Smart Connected Buildings Design Automation: Foundations and Trends*. Delft: now Publisher, p. 3.
- Moore, J., 2014. *Building automation systems: IoT meets facilities management*. [Online]  
Available at: <http://internetofthingsagenda.techtarget.com/feature/Building-automation-systems-Internet-of-Things-meets-facilities-management>  
[Accessed 25 March 2017].
- Nrk.no, 2014. *Yr.no er verdens femte største værnettsted*. [Online]  
Available at: <https://www.nrk.no/kultur/yr-er-verdens-femte-storste-vaerside-1.11815974>  
[Accessed 9 February 2017].
- OGC, 2017. [Online]  
Available at: <http://www.opengeospatial.org/>  
[Accessed 7 March 2017].
- Orenstein, D., 2000. *Application Programming Interface*. [Online]  
Available at: <http://www.computerworld.com/article/2593623/app-development/application-programming-interface.html>  
[Accessed 14 February 2017].
- Refsnes, J. E., n.d. *XML Basics - An Introduction to XML*. [Online]  
Available at: <http://www.xmlfiles.com/xml/>  
[Accessed 21 February 2017].

Rouse, M., 2016. *Internet of Things (IoT)*. [Online]  
Available at: <http://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT>  
[Accessed 25 March 2017].

Schneider Electric, 2014. *Enterprise Software Suites - Building Operation*. [Online]  
Available at: <http://www.schneider-electric.com/b2b/en/solutions/enterprise-solutions/solutions/enterprise-software-suites-building-operation/>  
[Accessed 7 March 2017].

Schneider Electric, 2016. [Online]  
Available at: <http://www.schneider-electric.com>  
[Accessed 13 February 2017].

Schneider Electric, 2017. *Smartstruxure solutions*. [Online]  
Available at: [http://www.schneider-electric.com/en/product-range/62111-smartstruxure-solution/189045889-automation-server-family?subNodeId=210024562en\\_WW#tabs](http://www.schneider-electric.com/en/product-range/62111-smartstruxure-solution/189045889-automation-server-family?subNodeId=210024562en_WW#tabs)  
[Accessed 8 February 2017].

Techterms, 2016. *API*. [Online]  
Available at: <https://techterms.com/definition/api>  
[Accessed 14 February 2017].

Trafiklab, n.d. *Vad är ett API?*. [Online]  
Available at: <https://www.trafiklab.se/vad-ar-ett-api>  
[Accessed 14 February 2017].

Tutorialspoint, 2017. *RESTful Web Services - Introduction*. [Online]  
Available at: [https://www.tutorialspoint.com/restful/restful\\_introduction.htm](https://www.tutorialspoint.com/restful/restful_introduction.htm)  
[Accessed 8 March 2017].

W3C, 2011. *Web Services Activity*. [Online]  
Available at: <https://www.w3.org/2002/ws/>  
[Accessed 14 February 2017].

Wang, S., 2015. *Intelligent Buildings and Building Automation*. Abingdon: Routledge.

Yr.no, 2016. *Fakta om Yr*. [Online]  
Available at: <http://om.yr.no/info/fakta/>  
[Accessed 9 February 2017].

Yr.no, 2016. *Vêrvarsel i XML-format*. [Online]  
Available at: <http://om.yr.no/verdata/xml/>  
[Accessed 9 February 2017].

	Format	Temperature	Precipitation	Humidity	Wind speed	Wind direction	Pressure	Sunrise/set	Cloud-cover
<b>Yr.no</b>	xml	Celsius	mm/h	%	mps	deg/direction	hPa	timestamp	variable
varsel_nu.xml									
forecast_hour_by_hour.xml									
forecast.xml			min/max						
<b>Foreca.com</b>	xml								
Entry		High/Low							
Standard		High/Low							
Premium		High/Low							
<b>Ilmatieteenlaitos.fi</b>	xml								
Open data									

Intervals by:	Minutes	Hours	Days
<b>Yr.no</b>			
varsel_nu.xml	7 (90)		
forecast_hour_by_hour.xml		1 (48)	(2)
forecast.xml		6 (216)	(9)
<b>Foreca.com</b>			
Entry			3
Standard		6	(5)
Premium		3	(14)
<b>Ilmatieteenlaitos.fi</b>			
Open data		1 (36)	(2)

**Comments**

The green areas indicate available parameters

Yellow text indicates that it is a paid service

**Forecasts intervals**

Intervals in minutes/hours/days (with the total amount in parentheses)

## References

Fetches 07.12.2016

Yr.no

<http://om.yr.no/verdata/xml/>

Foreca.com

<http://corporate.foreca.com/se/products-services/data/batchxml>

Ilmatieteenlaitos.fi

<http://en.ilmatieteenlaitos.fi/open-data-manual-forecast-models>