Construction of a provisioning system for IP-telephony boxes
Abstract
The point of this project is to cover a gap in today’s supply of provisioning systems for VoIP endpoint devices. Today the norm is that every manufacturer of endpoints builds their own system to provision their endpoints, but most operators use a vast verity of products to meet their customers’ needs. This system is based on open source software to allow adaptations to different environments and demands from customers and to require no license fee thus being affordable to the small size and midsized operators. This thesis is mainly an engineering type of thesis and the main bulk of the work has been put in developing the solution.
Table of content:

Abstract .................................................................................................................................................... i

Table of content: ..................................................................................................................................... ii

List of figures ........................................................................................................................................... iv

List of tables ............................................................................................................................................ iv

1. Introduction ......................................................................................................................................... 1

   1.1 Background .................................................................................................................................... 1

   1.2 Purpose .......................................................................................................................................... 1

   1.3 Goal ............................................................................................................................................... 1

   1.4 Limitations ..................................................................................................................................... 1

2. Theory .................................................................................................................................................. 2

   2.1 Protocols ........................................................................................................................................ 2

   2.1.1 PSTN - Public Switched Telephone Network .............................................................................. 2

   2.1.2 VoIP – Voice over IP .................................................................................................................... 2

   2.1.3 SOAP - Simple Object Access Protocol ....................................................................................... 3

   2.1.4 HTTP - Hypertext Transfer Protocol ............................................................................................ 3

   2.1.5 TFTP – Trivial File Transfer Protocol ........................................................................................... 4

   2.1.6 OPP – Owera Provisioning Protocol ........................................................................................... 4

   2.1.7 CWMP - CPE WAN Management Protocol ................................................................................. 4

   2.2 Languages ...................................................................................................................................... 6

   2.2.1 PHP – PHP:Hypertext Preprocessor ........................................................................................ 6

   2.2.2 Java ......................................................................................................................................... 6

   2.3 Database ........................................................................................................................................ 6

   2.3.1 MySQL ..................................................................................................................................... 7

   2.3 Authentication ............................................................................................................................... 7

   2.3.1 Basic Access Authentication ..................................................................................................... 7

   2.3.2 Digest Access Authentication .................................................................................................... 7

3. Pre-analysis .......................................................................................................................................... 8

   3.1 Protocols ........................................................................................................................................ 8

   3.2 Languages ...................................................................................................................................... 8

   3.3 Database ........................................................................................................................................ 9

4. Methodology ....................................................................................................................................... 9

   4.1 Circumstances ............................................................................................................................... 9

   4.2 Preparations .................................................................................................................................... 9
List of figures
Figure 1 – Displays the protocol stack for CWMP ................................................................. 5
Figure 2 – “Models tab” of the GUI .................................................................................. 11
Figure 3 – “Groups tab” of the GUI ................................................................................. 12
Figure 4 – “Box tab” of the GUI ..................................................................................... 13
Figure 5 – “Customer tab” of the GUI ............................................................................. 14
Figure 6 – “Edit Settings tab” of the GUI ....................................................................... 15
Figure 7 – “Add Settings tab” of the GUI ....................................................................... 16
Figure 8 – “Log tab” of the GUI ..................................................................................... 17
Figure 9 – “Server Settings tab” of the GUI ................................................................. 17
Figure 10 – First part of class diagram ............................................................................ 18
Figure 11 – Second part of class diagram ....................................................................... 20
Figure 12 – Third part of class diagram .......................................................................... 22
Figure 13 – Fourth part of the class diagram ............................................................... 23

List of tables
Table 1 - Comparison between different languages [6] ....................................................... 8
1. Introduction

1.1 Background
As the market for IP-telephony companies grows rapidly many new companies arise. Many of these companies need to configure the endpoint devices for every customer manually. This works good for a while but as the company grows the many devices will make the manual configuration very time consuming.

VoiceTech is one of those companies that starts to grow and need a way to automatically configure the devices. We decided to look more into this and found that there ware solutions to the problem, but those were very expensive and not an option for a small company. After some discussion we decided to develop a system that automatically can configure every device.

1.2 Purpose
The purpose of this thesis is to develop a provisioning system for automatic configuration of CPEs based on open source. After some research for existing provisioning systems we found that those that exist are very expensive or does not meet the requirements as a fully functional system. There is a need for a open source provisioning system for small and midsize companies.

1.3 Goal
To achieve a working provisioning system for CPEs, we’ll focus on Ping Communications ATA box model nr: NPA201E for testing. Building a backend database containing the configuration settings for different endpoint configurations. Building a web-based GUI (Graphical User interface) to edit or add configurations to the database and to manage the provisioning server.

Our goal is also to make the system flexible and easy enough to be expandable and adaptable by a third party to cover a broader spectrum of endpoint devices and services or to better suit their environment. The entire project will be based on public available third party software and products that at the present does not charge for usage. Some level of security in the authentication between the endpoints and the solution are to be maintained.

1.4 Limitations
We decided to not encrypt the data transmitted between the provisioning server and endpoint devices. Every protocol needed to fulfill the requirements may not be fully implemented, since it may not be time for that.
2. Theory

2.1 Protocols
This section briefly explains the protocols used or assessed later in the thesis.

2.1.1 PSTN - Public Switched Telephone Network
The traditional worldwide network for landline telephony is called the Public Switched Telephone Network (hence forth PSTN). The PSTN combines analog, digital and electromechanical data links to carry signals to its end users. The PSTN is a switched network which means it only needs active connection between the network endpoints while a call between them are active. While most connections between endpoints and switch are analog connections, between switches they are most commonly digital connections. However in the experimental beginning of telephone history the networks where meshed, which means the connections between all endpoints where permanently linked. As networks expanded beyond a few endpoints it quickly became apparent that this wasn’t very practical as the connections grew exponentially to the number of endpoints. The main function of the PSTN is to setup and tear down links between endpoints.

Within the PSTN there is a signaling network called Signal System 7 (SS7) which purpose is to handle the non voice data flow within the PSTN. SS7 has borrowed 4 layers from the Open System Interconnection (OSI) model: 1 Physical, 2 Data link, 3 Network and 4 Application which corresponds to the layers 1-3 and 7 in the OSI-model. The first three layers called Message Transfer Part (MTP) while the fourth contains Signal Connection and Control Part (SCCP) and a number of different user parts i.e. Telephone User Part (TUP), ISDN User Part (IUP) and Mobile Application Part (MAP).

In the PSTN the sound signal, electrical power and the handling of dialing are all functions in the switch not the endpoints. To add functionality the hardware in the switches or even the switches themselves often need to be replaced.

2.1.2 VoIP – Voice over IP
VoIP is a generally term that covers a number of different signaling protocols that delivers the service voice over IP, some of the most commonly used are Session Initiation Protocol (SIP), Media Gateway Control Protocol (MGCP) and ITU-T-H.323. VoIP uses the ordinary Internet protocol (IP), thus enabling it to use the internet or any other network using IP. Thus it’s simpler, cheaper and less time consuming to implement new services and adding functionality than in the PSTN. Adding functionality generally doesn’t require any change to the network hardware itself. In VoIP it’s the endpoint that initiates the connection to the switch; this enables the endpoints to be geographically unbound. As it is the endpoint that initiates the connection, it needs to be correctly configured in order to be able to establish a connection with the switch. In order to accomplish this most devices comes pre-configured to connect to the manufactures server and depending on the registered buyer of the endpoints MAC-address be forwarded to that specific providers/companies provisioning system. The only other way to configure endpoints is to manually configure each device which is both time consuming and ineffective. However most manufactures only deliver provisioning systems for their own products and most service providers uses an assortment of different manufactures depending on what their clients require.
2.1.3 SOAP - Simple Object Access Protocol

SOAP uses XML to provide a message structure for communication between applications, with the main goal of simplicity and extensibility. It is platform and language independent, which means that applications using different platform and language can still communicate using the SOAP protocol [18].

Example of a SOAP message [14]:

```xml
<soap:Envelope
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
   xmlns:cwmp="urn:dslforum-org:cwmp-1-0">
   <soap:Body>
     <cwmp:Request>
       <argument>value</argument>
     </cwmp:Request>
   </soap:Body>
</soap:Envelope>
```

The envelope element is the root element, with a few namespaces, of the SOAP message and is required. The body element is the actual data to be transmitted to the endpoint and may have multiple child elements, in this case the Request element which also has a child. Other than those elements there is also the optional header element, which could contain information needed for the application to process the message such as priority settings. There’s also the predefined element Fault which are a child element to the body element and should be used whenever there’s any faults in the SOAP message [18]. An example of a fault message [14]:

```xml
<soap:Envelope
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
   xmlns:cwmp="urn:dslforum-org:cwmp-1-0">
   <soap:Header>
     <cwmp:ID soap:mustUnderstand="1">1234</cwmp:ID>
   </soap:Header>
   <soap:Body>
     <soap:Fault>
       <faultcode>Client</faultcode>
       <faultstring>CWMP fault</faultstring>
       <detail>
         <cwmp:Fault>
           <FaultCode>9000</FaultCode>
           <FaultString>Upload method not supported</FaultString>
         </cwmp:Fault>
       </detail>
     </soap:Fault>
   </soap:Body>
</soap:Envelope>
```

2.1.4 HTTP - Hypertext Transfer Protocol

HTTP is located on the application layer of the OSI-model and is used for distributed, collaborative, hypermedia information systems. HTTP has been used since 1990 and the first version was HTTP/0.9, and the current version is HTTP/1.1. The first version of HTTP was a protocol for transferring raw data across the Internet, but the later versions have added functionality for non-ASCII data to be transferred. HTTP defines a set of methods, such as POST which is used to post data to a server [17].

Example of a HTTP POST message:

```
POST HTTP/1.1
Host: www.example.org
Connection: Keep-Alive
```
The above example sends a XML message to the host www.example.org.

2.1.5 TFTP – Trivial File Transfer Protocol
Is defined in the RFC1350, was named so because of the rudimentary function it was meant to fill. It’s built on top of UDP, as such it inherit some of the less desirable properties of UDP i.e. it’s a connectionless protocol with no guaranteed delivery of packets. Another drawback with TFTP is the lack of authentication functionality. The only thing TFTP is capable of is transferring files [3].

2.1.6 OPP – Owera Provisioning Protocol
OPP was developed by Owera. Owera describes the protocol as:
“OPP (Owera Provisioning Protocol) has been designed bottom-up to provide remote management or monitoring. The OPP protocol differs from TR-069 and other polling standards since it is auto NAT, bidirectional and real-time. Owera has designed this for targeting real-time monitoring, management of applications from mobile phones (over WLAN) and other nomadic devices. OPP can be used to extend functionality of other protocols (i.e. monitoring).” [1]

2.1.7 CWMP - CPE WAN Management Protocol
TR-069 describes the generic components of CWMP in an environment comprising of a CPE and an ACS. CWMP contains a framework of mechanisms and management functions even though many individual adaptations for specific services and devices exist. The primary capabilities for which CWMP are intended is according to TR-069:

- Auto-configuration and dynamic service provisioning
- Software/Firmware image management
- Status and performance monitoring
- Diagnostics

In CWMP functions for getting and setting specific endpoint parameters are included, this makes it ideal for group based or role based provisioning.
The CWMP has a number of messages used to manage the endpoint device, examples of those and 
the most used are Inform, InformResponse, GetParameterValues and SetParameterValues. A session 
always starts with the endpoint sending an Inform message, though the server can force the device 
to start a session. An example of the Inform message are seen below:

```xml
<soap:Envelope xmlns:soap=http://schemas.xmlsoap.org/soap/envelope/
xmlns:soapenc=http://schemas.xmlsoap.org/soap/encoding/
xmlns:cwmp=urn:dslforum-org:cwmp-1-0
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xmlns:xsd=http://www.w3.org/2001/XMLSchema
xmlns:urn=dslforum-org">
  <soap:Header>
    <cwmp:ID soap:mustUnderstand="1">ubicom_tr069_id_1</cwmp:ID>
    <cwmp:NoMoreRequests>1</cwmp:NoMoreRequests>
  </soap:Header>
  <soap:Body>
    <cwmp:Inform xmlns:cwmp="urn:dslforum-org:cwmp-1-0">
      <DeviceId>
        <Manufacturer>Ping Communication</Manufacturer>
        <OUI>002194</OUI>
        <ProductClass>NPA201E</ProductClass>
        <SerialNumber>0021940012FC</SerialNumber>
      </DeviceId>
      <Event soapenc:arrayType="cwmp:EventStruct[4]">
        <EventStruct>
          <EventCode>4 VALUE CHANGE</EventCode>
          <CommandKey></CommandKey>
        </EventStruct>
        <EventStruct>
          <EventCode>1 BOOT</EventCode>
          <CommandKey></CommandKey>
        </EventStruct>
        <EventStruct>
          <EventCode>M Reboot</EventCode>
          <CommandKey></CommandKey>
        </EventStruct>
        <EventStruct>
          <EventCode>0 BOOTSTRAP</EventCode>
          <CommandKey></CommandKey>
        </EventStruct>
      </Event>
    </cwmp:Inform>
    <MaxEnvelopes>1</MaxEnvelopes>
    <CurrentTime>0001-01-01T00:00:06</CurrentTime>
  </soap:Body>
</soap:Envelope>
```
After this message the server answers with an InformResponse message [14].

2.2 Languages
This section briefly explains the strong points of the programming languages used in the thesis.

2.2.1 PHP – PHP:Hypertext Preprocessor
PHP is an open source server-side scripting language which is especially suited for Web development and can be embedded into HTML. PHP scripts are executed on the server and translated into HTML code, which lets the developer write dynamically generated web pages. It runs on several different platforms and it’s syntax are similar to C, Java and Perl which makes it easy to learn for developers with experience in any of these languages [4].

2.2.2 Java
Java is an open source class based object oriented programming language. It’s designed to have as few implementation dependencies as possible. Programs developed in Java are not bound to the developing platform but can be run on almost any platform [6].

2.3 Database
This section briefly explains the DBMS used in the thesis.
2.3.1 MySQL
MySQL is an open source relational database management system available for over 20 platforms. MySQL has support for many different programming languages including C, C++, C#, Java, PHP, Perl, Python, and Ruby [10].

2.3 Authentications
This section briefly explains the authentication methods used or assessed during the rest of the thesis.

2.3.1 Basic Access Authentication
Basic Access Authentication operates with challenge-response model. The server challenges the client to authenticate for a specified realm, the client then respond by sending the username and password as an base64 encoded string in the form “username:password”. The server compares the response with the accepted responses [12]. Base64 is a term used to describe a representation of information in 65-character subset that represents a-z, A-Z, 0-9, = and two optional characters. The = is used as padding to achieve the desired length of information [13].

2.3.2 Digest Access Authentication
Digest authentication uses hashing (default is MD5) to not send passwords in clear text. Digest Access Authentication operates by challenge-response communication and a nonce values to get a unique hash for each authentication of a connection. The client response with a hash based on username, password, nonce value, HTTP method and URI. The password needs to be prearranged and both the client and the server need to have the prearranged password. The Digest Access Authentication also has a number counter value that keeps tracks of the number of messages sent so far by the client [12].
3. Pre-analysis

3.1 Protocols
The test endpoint supports three different provisioning protocols: TFTP, CWMP and OPP.

The solution implements CWMP as defined in TR-069 as it’s the industry standard and the only protocol that lives up to our requirements. TFTP lacks the needed authentication functions and OPP are as of this date not an open protocol.

3.2 Languages

<table>
<thead>
<tr>
<th>Feature</th>
<th>.Net</th>
<th>Java</th>
<th>Cold Fusion MX</th>
<th>PHP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compiled Code – Increases website speed (precompiled is the fastest)</td>
<td>Yes – both precompiled and dynamically compiled when a page is requested</td>
<td>Yes – both precompiled and dynamically compiled when a page is requested</td>
<td>Yes – dynamically compiled when a page is requested</td>
<td>No – a 3rd party accelerator can be used to increase performance but it is not installed on most shared hosting servers.</td>
</tr>
<tr>
<td>Scripted Language – results in poor website performance</td>
<td>No</td>
<td>No</td>
<td>Somewhat</td>
<td>Yes – a 3rd party accelerator can be used to increase performance but it is not installed on most shared hosting servers.</td>
</tr>
<tr>
<td>Object Oriented – Increases the ability for code reuse and provides enhanced features as well as reduced development time; since code is more reusable, results in fewer bugs that can be discovered by any client and fixed for everyone; encourages developers to write more maintainable code.</td>
<td>Yes</td>
<td>Yes</td>
<td>Somewhat</td>
<td>No</td>
</tr>
<tr>
<td>Supported Development Languages – easier to find developers</td>
<td>C++, C#, Visual Basic.NET, Jscript.NET, Python, Perl, Java (J#), COBOL, Eiffel, Delphi – 25 languages supported currently</td>
<td>Java</td>
<td>CFML and CFScript</td>
<td>PHP</td>
</tr>
<tr>
<td>Browser Specific HTML Rendering – different HTML is automatically sent to IE than to Netscape, reducing incompatibility issues</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Open Source</td>
<td>No</td>
<td>Yes</td>
<td>Somewhat</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 1 - Comparison between different languages [6]

The provisioning server is implemented in Java and the GUI in PHP. Compared to PHP, Java offers a more wide range of functions while PHP is more centered on web development. Java also offers a wider range of developing tools which makes debugging easier and cuts down on developing time.
3.3 Database
Due to the open source requirement and the experience of the developers the solution is implemented in MySQL. There are other open source database management systems such as PostgreSQL but the project team lacks experience with it.

4. Methodology

4.1 Circumstances
The initial lab environment consisted of a Ping Communications ATA box model nr: NPA201E and a laptop. This however resulted in communication errors that manifested itself by the box only randomly requesting information from the laptop. As a result testing could only be performed a few times a day. Initially the cause of this error was unknown, but after contact with the manufacturer of the box it was quickly determined to be due to the box and laptop being in the same subnet. As a result the server was moved to a computer at the VoiceTech office. The final setup consisted of two virtual servers at VoiceTech running the web server and backend database, the endpoint running anywhere with an internet connection and a laptop remotely connected to the servers.

4.2 Preparations
The project started with the defining of a class diagram and the creation of a simple testing module. This module was listening for HTTP POST requests, in this case a SOAP message and tried to answer with a POST response. The endpoint was sending an Inform message, defined in TR-069, and the server was responding with an InformResponse message.

4.3 Packet listening
To make troubleshooting and overview of the communication between the endpoint and server the packet listening program Wireshark was installed. Wireshark is released under the GPLv2, and thus within the limitations of the project. Wireshark was also used to listen in on a working provisioning communication between the manufacturers provisioning server and the endpoint. Once this was achieved a fully verified mapping of the communication established.

4.4 Communication module
The communication module is designed to coincide with the standards defined in [14].

For more information see: 5.7 The code structure.

4.5 Database

4.5.1 Design
The database consists of the following nine tables:

1. Box(MAC, GroupID, CustomerID, ModelID, Username, Password)
2. Customer(ID, Name)
3. Group(ID, Name)
4. Model(ID, Name, OUI, Manufacturer)
5. Settings(ID, ModelID, Setting, Flags)

6. Settings_box(BoxID, SettingID, Value)

7. Settings_customer(CustomerID, SettingID, Value)

8. Settings_group(GroupId, SettingID, Value)

9. Settings_standard(ModelID, SettingID, Value)

10. Log(ID, IP, Timestamp, Level, MAC, Message)

For a relationship diagram between the tables see Appendix A.

4.5.2 Implementation
The database is implemented in MySQL via phpMyAdmin which is an open source tool written to handle administration of MySQL [11]. The database utilizes the latin1_general_ci collation. The values in the password column in the Box table are encoded using AES to avoid storing sensitive information in plain text.

4.6 Authentication

4.6.1 Comparison
Compared to Basic, Digest Access Authentication never sends the password in clear text. The digest could in some scenarios be used to obtain the same transaction again, however most often this is prevented by the server using a new nonce (number used once). The number counter also helps prevent the reuse of eavesdropped digests. In a security context Basic Access Authentication is not recommendable due to the many different vulnerabilities it generates when sent over a network, unless the traffic is encrypted on a different layer. The Basic does have an advantage over Digest Access Authentication in a performance context as it requires less information to be encoded and also uses a much simpler model to encode the information [12].

4.6.2 Implementation
The solution authenticates with Digest Access Authentication. The digest authentication model was chosen as it best corresponds with the wanted solution. As the solution uses the MAC of the endpoint to verify with the database if the supplied user name and password are correct.

4.7 GUI

4.6.1 Design
The design is kept simple and without any advanced extra functions. The group, model and box tabs contain a basic interface to create and manage information stored in their corresponding database tables. The setting tabs is used to link settings for group, box, model etc. The server settings tab contains the settings for the provisioning server, such as network and log settings.

4.6.2 Implementation
The GUI is web based and written in PHP. Every page consists of a master page and the page content. In the top left corner the online/offline status of the server is displayed.
4.6.3 Models tab

Add model

Model name: 
OUI: 
Manufacturer: 

Save  Reset

All models

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>OUI</th>
<th>Manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NPA201E</td>
<td>002194</td>
<td>Ping Communication</td>
</tr>
<tr>
<td>23</td>
<td>RGW208EN</td>
<td>002194</td>
<td>Ping Communication</td>
</tr>
</tbody>
</table>

Edit model

Edit model RGW208EN
ID: 23
Model name: RGW208EN
OUI: 002194
Manufacturer: Ping Communication
Save  Cancel

Figure 2 – “Models tab” of the GUI

List of functions on this tab:

- Add a new Model of endpoint. To do this the model name, manufacturer and OUI must be known.
- List all the available models already entered, including their OUI and the name of the manufacturer.
- Edit an already created model.
- Reset the model part of the database.
4.6.4 Groups tab

Server online

<table>
<thead>
<tr>
<th>Models</th>
<th>Groups</th>
<th>Box</th>
<th>Customer</th>
<th>Edit settings</th>
<th>Add settings</th>
<th>Log</th>
<th>Server settings</th>
</tr>
</thead>
</table>

Add group

Group name:

Save
Reset

All groups

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>Test</td>
</tr>
<tr>
<td>34</td>
<td>Test2</td>
</tr>
</tbody>
</table>

Edit Delete Reset

Figure 3 –“Groups tab” of the GUI

The “Groups tab” is used to manage the different groups of configurations.

List of functions on this tab:

- Create new group.
- Edit existing group.
- Delete existing group.
- Reset the “Groups” table of the database.
4.6.5 Box tab

The Box tab is used to add new, and edit or view existing boxes and assigning their configurations.

List of functions on this tab:

- A added box can contain the parameters:
  - The MAC address of the new box is supplied when ordering the endpoint.
  - The ModelID of the box can be found under the Models tab.
  - The GroupID can be found under the “Groups” tab.
  - The CustomerID can be found on the “Customer” tab.
  - Username and password is used to authenticate the box once it requests a provisioning.
- List of boxes and configuration IDs.
- Edit existing boxes.
- Delete existing boxes.
- Reset the Box table of the database.
4.6.6 Customer tab

Server online

| Models | Groups | Box | Customer | Edit settings | Add settings | Log | Server settings |

Add customer

ID: 
Name:  
Save  Reset

All customers

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Micke</td>
</tr>
<tr>
<td>5</td>
<td>Test</td>
</tr>
</tbody>
</table>

Edit  Delete  Reset

Figure 5 - "Customer tab" of the GUI

On the customer tab configuration settings for specific customers are assigned.

List of functions on this tab:

- Add new customer ID and assign name to the ID.
- List the existing IDs and customers.
- Edit existing customers
- Delete customers.
- Reset the customer table of the database.
4.6.7 Edit Settings tab

Server online

<table>
<thead>
<tr>
<th>Models</th>
<th>Groups</th>
<th>Box</th>
<th>Customer</th>
<th>Edit settings</th>
<th>Add settings</th>
<th>Log</th>
<th>Server settings</th>
</tr>
</thead>
</table>

List settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Flags</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>InternetGatewayDevice.ManagementServer.PeriodicInformEnable</td>
<td>RW</td>
<td>1</td>
</tr>
<tr>
<td>InternetGatewayDevice.ManagementServer.PeriodicInformInterval</td>
<td>RW</td>
<td>86400</td>
</tr>
<tr>
<td>InternetGatewayDevice.ManagementServer.URL</td>
<td>RW</td>
<td><a href="http://example.org">http://example.org</a></td>
</tr>
<tr>
<td>InternetGatewayDevice.Services.VoiceService.1.PhyInterface.2.X_OWERA.COM_Impe</td>
<td>RW</td>
<td>270~750[150m]</td>
</tr>
<tr>
<td>InternetGatewayDevice.Time.DaylightSavingsUsed</td>
<td>RW</td>
<td>1</td>
</tr>
<tr>
<td>InternetGatewayDevice.Time.Enable</td>
<td>RW</td>
<td>1</td>
</tr>
<tr>
<td>InternetGatewayDevice.Time.NTPServer1</td>
<td>RW</td>
<td>time.nist.gov</td>
</tr>
<tr>
<td>InternetGatewayDevice.Time.X_OWERA.COM_TimeZone</td>
<td>RW</td>
<td>3600</td>
</tr>
<tr>
<td>InternetGatewayDevice.X_OWERA.COM.Console.Enable</td>
<td>RW</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 6 - "Edit Settings tab" of the GUI

List of functions on this tab:

- Select the model, group, customer or box configuration that should be edited.
- The available settings will be listed in an editable table as shown in [Figure 6].
- Input the desired changes directly into the shown settings table.
4.6.8 Add settings

Server online

<table>
<thead>
<tr>
<th>Models</th>
<th>Groups</th>
<th>Box</th>
<th>Customer</th>
<th>Edit settings</th>
<th>Add settings</th>
<th>Log</th>
<th>Server settings</th>
</tr>
</thead>
</table>

**Add settings**

Choose a model

Settings:

Save

**Figure 7 - "Add Settings tab" of the GUI**

List of functions on this tab:

- Choose the model to add setting for.
- Input the setting to add and press save.
### 4.6.9 Log

**Server online**

<table>
<thead>
<tr>
<th>Models</th>
<th>Groups</th>
<th>Box</th>
<th>Customer</th>
<th>Edit settings</th>
<th>Add settings</th>
<th>Log</th>
<th>Server settings</th>
</tr>
</thead>
</table>

**List log**

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>IP</th>
<th>Level</th>
<th>MAC</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010-05-12 15:27:06</td>
<td>77.105.198.152</td>
<td>INFO</td>
<td>0021940012FC</td>
<td>Server sent message: SetParameterValues</td>
</tr>
<tr>
<td>2010-05-12 15:27:05</td>
<td>77.105.198.152</td>
<td>INFO</td>
<td>0021940012FC</td>
<td>Server sent message: GetParameterValues</td>
</tr>
<tr>
<td>2010-05-12 15:27:04</td>
<td>77.105.198.152</td>
<td>INFO</td>
<td>0021940012FC</td>
<td>Server sent message: InformResponse</td>
</tr>
<tr>
<td>2010-05-12 15:27:04</td>
<td>77.105.198.152</td>
<td>INFO</td>
<td>0021940012FC</td>
<td>Server sent Inform message with a successful authentication</td>
</tr>
<tr>
<td>2010-05-12 15:24:43</td>
<td>77.105.198.152</td>
<td>INFO</td>
<td>0021940012FC</td>
<td>Server sent empty message, ending the transaction</td>
</tr>
</tbody>
</table>
CWMPCode: "9002" CWMPString: "internal error" |
| 2010-05-12 15:24:42 | 77.105.198.152 | INFO  | 0021940012FC | Server sent message: SetParameterValues |
| 2010-05-12 15:24:42 | 77.105.198.152 | INFO  | 0021940012FC | Server sent message: InformResponse |
| 2010-05-12 15:24:42 | 77.105.198.152 | INFO  | 0021940012FC | Server sent Inform message with a successful authentication |
| 2010-05-12 15:18:55 | 77.105.198.152 | INFO  | 0021940012FC | Server sent empty message, ending the transaction |
CWMPCode: "9002" CWMPString: "internal error" |
| 2010-05-12 15:18:52 | 77.105.198.152 | INFO  | 0021940012FC | Server sent message: SetParameterValues |
| 2010-05-12 15:18:48 | 77.105.198.152 | INFO  | 0021940012FC | Server sent message: GetParameterValues |

**Figure 8** - "Log tab" of the GUI

Here it's possible to view the messages sent to and from the server.

### 4.6.10 Server settings

**Server online**

<table>
<thead>
<tr>
<th>Models</th>
<th>Groups</th>
<th>Box</th>
<th>Customer</th>
<th>Edit settings</th>
<th>Add settings</th>
<th>Log</th>
<th>Server settings</th>
</tr>
</thead>
</table>

**Server settings**

- **IP:** localhost
- **Port:** 5001
- **Realm:** torstrailm@host.com
- **qop:** auth
- **PrivateKey:** CWMP_Server

**Figure 9** - "Server Settings tab" of the GUI

List of Functions on this tab:

- Setting the connection properties of the server.
4.7 The code structure

The code structure is described by the following figures and descriptions:

Figure 10 - First part of class diagram
The first part of the class diagram consists of the following classes:

- Main is the main class holding the main method that creates a Server class and starts the server.
- Server extends the Java ServerSocket, to simplify the configuration of connection settings.
- Logg holds the static method `writeLog` which records the server messages to the log table of the database.
- I_Database is an interface describing the communication to the database.
- Database is an implementation of I_Database.
- ClientThread handles the client messages and initiates the appropriate action on the server side.
- DigestAuthentication authenticates a client using the digest access authentication.
- Hex holds a static method to encode a string into hexadecimals.
- Message is an abstract class which function is to convert client messages into objects. It holds the abstract methods `parseBody` and `createBody` that all subclasses has to implement. Subclasses to Message are messages that can be sent to or from the server. Each message has its corresponding response message to report the response of the message.
- GetPRCMethods are used by both client and server to check which methods are supported.
Figure 11 - Second part of class diagram
The second part of the class diagram consists of the following classes:

- SetParameterValueValues are used by the server to set parameters on the client. It holds a list of ParameterValueStruct that represents one parameter.
- GetParameterValueValues are used to get the values of a collection of parameters.
- GetParameterNames are used to get all parameter names within a path.
- SetParameterAttributes are used to set attributes for parameters.
- SetParameterAttributesStruct It holds a list of where the values are stored.
- GetParameterAttributes are used to get the attributes of the chosen parameters.
- AddObject are used to add custom parameters.
- DeleteObject are used to remove parameters.
- Download initiates a client side download from a specified location.
- Reboot are used to force a reboot of the client.
Figure 12 - third part of class diagram
The third part of the class diagram consists of the following classes:

- Inform are used by the client to initiate a connection and holds, a DeviceldStruct that holds all the information needed to identify the client, and an EventStruct containing the reason the message was sent.
- TransferComplete are used by a client to inform whether a file transfer was successful or not.
- AutonomousTransferComplete are used by same principal as TransferComplete but for autonomous transfers.
- GetQueuedTransfers is a deprecated predecessor to GetAllQueuedTransfers and are used to get the status of progressing transfers.
- ScheduleInform are used to schedule an Inform message.
- SetVouchers are used to set Vouchers.
- GetOptions are used to get information about the supported options of the client.
- Upload initiates a client side upload to a specified location.
- FactoryReset are used to reset the client to factory defaults.

![Diagram](image)

Figure 13 - forth part of the class diagram

The fourth part of the class diagram consists of the following classes:

- GetAllQueuedTransfers are used to get a list of all transfers in progress.
- Kicked are used when the client requests a redirect to a different server.
- RequestDownload are used by the client if it desires a download.
- Fault messages are used to relay a message containing information about an error.
5. Post analysis

5.1 Adding new types of endpoints
The test was performed by adding the functionality required to provision an endpoint device of the type RGW208EN, which is also made by Ping Communication and therefore is similar but not identical to the original test endpoint NPA201E. The test was successful at the very first attempt tried and was completed by only using the solutions GUI. The test lacks in that it was done on a similar device but still shows that as long as the device implements the TR-104 directions for included parameters, a fully operational provisioning can be achieved.

5.2 Firmware distribution
Functionality for firmware distribution is also defined in the TR-069 amendment 2 [14], and was added due to a strong demand from VoiceTech. As the framework and basic functionality was already in place this was implemented into the solution in 4 hours and tuned in and tested to work with the VoiceTech environment in another 4 hours. The firmware distribution function utilizes the Download message with the arguments defining where and how the client can download the new firmware.

5.3 Adaptations to environment
The general solution uses its own database in order to be self proficient, but VoiceTech very strongly felt that the solution should be integrated against their existing customer database in order to prevent the same information being stored redundantly in dual databases. Due to the built-in flexibility and adaptability this too was achieved in a relatively simple manner by using the existing GUI functions but changing the backend database command being processed on the server. This adaptation later was taken another step by enabling the existing customer GUI to interact with the server and thus achieving a full integration to the pre-existing VoiceTech environment. The only functions which still require the built-in database are the pure settings per model tables. These tables can of course also be migrated to an external database but this will require more work than this thesis permit.
6. Discussion

6.1 Reaching the goals
The code is structured according to the class diagram, the custom classes have been logically named to make the code easier to understand for future development by third parties. There are a lot of areas in the solution that can be developed to add further functionality. Through the database interface it’s easy to adapt the solution to work against for instance a customer database. In fact the solution was adapted by VoiceTech to do just that, it took less than a day. The adaptability and flexibility in the solution is one of its strongest benefits and was also one of the main goals for the project. From the security standpoint the solution lives well up to the goal of achieving some authentication, and even exceeds them in the fact that the authentication information is stored in an encrypted form. The GUI covers the required functions to serve a fully operational provisioning system for a small or midsized company. We also added functionality to distribute firmware to the endpoints, but as it was not part of the original scope we choose to not document it. The system has already been taken into operational status and performs well. The code could be optimized for better performance but with the time span of the project we deemed it less important. One of the biggest issues left to do is further testing of the system, such as a stress test and testing of more endpoint devices from other manufacturers.

6.2 Future expandability features
A known weakness with the system is that the solution does not have any event triggers on fault messages but simply logs them and then depends on manual handling of all errors.

Worth thinking about is the fact that this system lacks predefined adaptation and interaction interfaces and thus requires some programming knowledge to fully interact with other custom applications.

The system is also lacking in that there is no login to the GUI, the server is meant to operate behind a firewall using NAT and should not be public accessible on the port chosen for the GUI. Securing the GUI was not within the original scope/goal of the thesis but this is a critical feature for many possible future users and should be seen as the next step.

SSL/TLS encryption of the communication between endpoint and server. Encryption is to ensure no information is intercepted and SSL/TLS encryption is the only mentioned encryption method in the CWMP definition.

Time optimization of the code and database. Due to lack of time we didn’t optimize to the extent that we wanted to, which might cause the system to experience performance issues in environments that are much bigger than Voicetech.
7. Acronyms

PHP - PHP: Hypertext Preprocessor
HTTP – HyperText Transfer Protocol
SOAP – Simple Object Access Protocol
ATA – Analogy telephony adapter
TR-069 – Technical report 069
CWMP – CPE WAN Management Protocol
CPE – Customer-premises equipment
WAN – Wide Area Network
LAN – Local Area Network
PSTN – Public Switched Telephone Network
OSI – Open System Interconnection
IP – Internet Protocol
TCP – Transport Control Protocol
UDP – User Datagram Protocol
MAC – Media Access Control
VoIP – Voice over Internet Protocol
TFTP – Trivial File Transfer Protocol
OPP – Owera Provisioning Protocol
RFC – Request For Comments
ACS – Auto-Configuration Server
HTML – Hyper Text Markup Language
RDBMS – Relational Database Management System
DBMS – Database Management System
GPLv2 – GNU General Public License version 2
MD5 – Message-digest algorithm 5
Nonce – Number used once
URI - Uniform Resource Identifier
OUI – Organizationally Unique Identifier
8. References

8. Learning Java, 3rd Edition By Jonathan Knudsen, Patrick Niemeyer
18. http://www.w3.org/TR/soap12-part1
19. http://www.w3schools.org
Appendix A

ERR diagram of the relationships between each table.