IT 08 032
Examensarbete 30 hp
September 2008

Contact Center with Mobile Agents

Niklas Burvall

Institutionen för informationsteknologi
Department of Information Technology

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Contact Center with Mobile Agents

Niklas Burvall

A Contact Center assists a company with customer relations such as support and telemarketing using telephones but also with email, SMS and chat. Incoming calls are routed by the contact center to the best match according to the polices set up and the queue status. In a contact center context the person handling the calls is called an agent. It would be beneficial to be able to reach agents which are not at the contact center but at home or in the field.

This master thesis investigates how such a mobile agent can be connected to a contact center with respect to security, handling more than just voice interactions (i.e. chat and email) and requirements on network infrastructure from a mobile perspective. A prototype client is developed using Java ME connecting to a Genesys contact center, describing the requirements a client has on network, mobile device and security. The results show that eventhough a mobile agent is possible using todays technologies, some features still need to be added to the mobile phones API, such as better call control and keeping focus during incoming calls.

Handledare: Ali Nouri, Jonas Flygare
Ämnesgranskare: Olle Gällmo
Examinator: Anders Jansson
IT 08 032
Tryckt av: Reprocentralen ITC

Kundtjänst med mobila agenter

Sammanfattning

En kundtjänst hjälper ett företag med kundkontakter vid t.ex. kundsupport och telemarketing men främst via telefon samt även via e-post, SMS och chatt. Inkommande telefonsamtal tas om hand av systemet och skickas till den person som verkar mest lämplig att besvara samtalet. Extra information som kan hjälpa systemet att ta rätt beslut kan samlas in med hjälp av att kunden t.ex. knappat in sitt kundnummer eller frågor besvarade i talsvar.

De personer som hanterar samtalen, s.k. agenter, loggar in till en kundtjänst som är konfigurerat med deras individuella färdigheter, språkkunskaper och annan information som kan hjälpa kontaktcentret att styra samtalen bättre. Agenterna förses med extra information när samtalet kopplas och den informationen kan även skickas till andra system för att automatiskt visa bakgrundsinformation vilken kan behövas för att möta kundens behov. Kundtjänster är ofta installerade direkt på plats vid företaget för att kunna tillhandahålla access till företagets andra system och för att kunna vara nära en telefonväxel osv. Många kundtjänster använder också externa agenter, dvs. agenter som arbetar utanför kontoret men som kan behövas för andra/tredje linjens support. Dessa agenter kan använda en VPN uppkoppling till företagets interna nät för att kunna logga in till sin kundtjänst.

I vissa affärsverksamheter kan det vara till stor nytta att kunna nå potentiella agenter som av olika anledningar inte kan komma åt företagets kontaktcenter. Idag kan dessa agenter nås från ett kontaktcenter men de kan inte tillgodogöra sig bakgrundsinformation, uppdatera information och kontaktcenter kan inte se om agenten t.ex. är upptagen.

Det här examensarbetets uppgift är att undersöka hur en mobil agent potentiellt kan kopplas ihop med ett kontaktcenter med avseende på säkerhet, möjligheter att hantera mer än bara röstsamtal (t.ex. chat, email osv) och krav på nätverksinfrastrukturen. En prototyp-applikation för mobiltelefoner har utvecklats där det är möjligt att hantera inkommande samtal och se tillhörande kunddata.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A contact center is an institution that handles a companies customer interactions such as telephone calls, email, chat, etc. The contact center platform is an application framework to handle customer interaction flows and directing each interaction to the most suitable person. This person is in a contact center context called an agent. Agents working in the contact center log in to the framework which is configured with the agent skill, department and other information that is used by the framework to correctly route interactions.

Agents are provided with interaction information when the interaction is delivered and may also have that information sent to legacy systems automatically in order to provide the agent with the background information needed to meet customer needs. Contact centers are usually deployed at the various company locations in order to provide access to legacy systems and also due to physical restrictions such as proximity to a PBX and other support systems.

Many contact centers also use external agents, that is agents that may work out of office but may be needed for second/third line handling of interactions. Such an agent might use a VPN connection and IP telephony in order to log in to the contact center framework. For certain businesses it would be beneficial to also be able to include the potential agents with the skills needed to handle customer interactions but who are unable to access the framework wirelessly. Today those agents may get interactions sent to them from the contact center but they are unable to receive background information about the interaction or to update the information. A mobile agent is an agent that has a mobile terminal capable of simultaneously handling the interaction and the data traffic to and from the contact center framework. Additionally the mobile agent software should run on a terminal small enough to be carried with the agent at all times, for example a mobile phone using 3G or a PDA using WiFi.

1.1 Background

1.1.1 TeliaSonera

TeliaSonera provides fixed voice, broadband and mobile services in the Nordic and Baltic countries. The master thesis was ordered by the Customer Integrated Solutions (CIS)

group located at TeliaSonera Uppsala. CIS specialices in contact center solutions with the two main products being Genesys from Alcatel-Lucent and CallGuide which is developed in house.

1.1.2 Genesys

Genesys is a contact center software platform from Alcatel-Lucent. Using the APIs provided by Genesys it is possible to extend the platform by creating customized clients and other software to enhance the contact center for a specific customer. The work in this thesis is based on Genesys and its interfaces.

1.2 Related Work

The concept of having a mobile agent in a contact center is not new but there are not that many commercial products available. During this thesis only one other mobile client software was evaluated. The platform for the software is Java ME and is distributed to the clients via a provisioning server using Over-The-Air technology. The mobile phone model is identified and a custom version specifically tailored for that user and mobile phone is sent out. If there is no match for the specific phone model a generic version is sent. The provisioning server is run on the Tomcat J2EE servlet server. During initial tests at TeliaSonera some issues with connecting to the server were discovered. The client works well using an emulator but on a real device and on a real network the session with the server is interrupted. The client sets up two connections with the server. One for incoming messages from the server and one socket for the clients communication with the server. The initial problems experienced using this product may have a possible in that the real mobile network uses a proxy server between the mobile phones and the internet. This proxy server can cause firewall problems which are hard to predict and debug.

Chapter 2

Problem Description

The purpose of this thesis work is to examine how to include mobile agents in a contact center. A prototype application that can handle inbound interactions is to be built in a lab environment at TeliaSonera.

2.1 Goals

The following aspects of a mobile contact center agent should be taken in consideration.

1. Alternative solutions: Define and prioritize between possible solutions to enable mobile agents in a contact center. Pros and cons? (GPRS/EDGE/WLAN/VOIP etc)
2. Features:
   - Enabling mobile agents to access common contact center functions, documenting features and limitations in the process.
   - Requirements on infrastructure, applications and protocols.
   - Inventory of available terminals/client applications.
3. Security: Address and suggest solutions to security issues such as preserving the integrity of business and customer data when the mobile agent is accessing the contact center functions via public voice and data networks.
4. Network integration: How is the integration of data traffic with the 3G networks done? What possibilities are there?
5. Multimedia: Is it possible for Mobile Agents to handle multimedia, e-mails, IM/chat) within the contact center platform framework. How could this be done? What is needed?

Secondary goals:

1. Network dimensions: What bandwidth is necessary based on traffic? Examine traffic models and present a basis for network recommendations.
2. Economy: Make a model for calculating investments and maintenance costs.

3. Market inventory: Are there any existing contact center installations using mobile agents? If so, how do those use Mobile Agents, and/or what are the limitations and benefits with those solutions?

2.2 Restrictions

Restricting the goals of this master thesis is necessary in order to avoid trying to solve problems all at once. The first restriction is that the contact center platform used for the mobile agent will be Genesys from Alcatel-Lucent. Secondly the mobile agent will only be developed and tested on one platform. The client will only support voice interactions in the first prototype other contact center functions and interaction types will be implemented if there is time. Due to time restrictions only the first five goals listed under the problem description are prioritized.

Chapter 3

Wireless Technology

Connecting to a contact center using a desktop or laptop computer is possible with the use of Virtual Private Networks (VPN). A user can connect using the companies standard contact center client and could in that sense be seen as a mobile agent. This thesis will look into making the agent truly mobile so that the client software should be able to run on a device small enough to be carried around by the user at all times, such as a mobile phone or PDA. This chapter will investigate the requirements a mobile agent will have on the wireless network.

3.1 Mobile Networks

Mobile phones currently have many ways to connect to the Internet, such as GSM, GPRS, 3G and WLAN — each with its pros and cons. The first generation of phones capable of connecting to the internet used the Wireless Application Protocol (WAP) which basically was a dialup connection to the ISP and worked like a normal phone call. The introduction of General Packet Radio Services (GPRS) allowed users to share the radio resources and not putting any load on the network when not sending or recieiving packets. In order for a GSM GPRS phone to handle both packet data and voice at the same time two transceivers are needed. Most phones have only one transceiver capable of handling either packet or voice. A network connection to the Internet either blocks the incoming call or disconnects the data session [22]. In a contact center context neither of these two options are acceptable. If the mobile agent client blocks incoming calls the entire purpose of the mobile agent platform becomes severely limited. In the case of the network connection dropping on an incoming call the client will have to reconnect after each interaction which potentially can become very annoying for the user.

One of the advantages the 3G network offers over the older GSM network is that it is possible to have both a voice call and still have a network connection up at the same time. The data rates of 384Kbps is more than enough to handle communication with the contact center for all media types (chat and email). Turbo 3G (HSDPA) upgrades done to the 3G network will increase the download capabilities of the client and enable use of more richer media.

An alternative to using a 3G connection, which without a good data plan can cost up to 2 euro per megabyte, is to use a WLAN connection which has become a common feature on high end devices such as smartphones and PDAs. The benefits of having a network connection without restrictions on how much data can be transferred is damp-

ened by the increased power usage and limited range of the wireless network. Handling the hand over from WLAN to 3G is a very complex and time consuming process and outside the scope of this master thesis.

The target preferred network should be 3G but since a WLAN connection works equally well from the client perspective it is up to the end user to decide on how the client should connect to the internet. Other technologies such as WiMAX does not currently have the widespread deployment as that of 3g and will for this reason not be considered.

One of the big problems with using the network connection of a mobile phone is that the standby time drops from weeks to hours. Having an internet connection up is equivalent to a voice call. In order to get the most out of the battery a contact center client should be very conservative with its resources and only send data when needed [22].

3.2 VoIP and SIP

The trend in the telecom industry is to migrate from standard telephone networks to Internet based technologies. Genesys and other contact center platforms supply a VoIP alternative to the standard PBX solution. This opens up new possibilities for a mobile agent client to control the call using SIP - which is a protocol used by VoIP clients for opening sessions. Using SIP to just control where the call is routed or using a SIP client for the voice interaction. SIP also supports other interaction types such as text chat. The problem is that a good data plan is needed since costs associated with a 3G connection usually are very high. Mobile operators also do not like the idea of giving free voice calls to their users and in some cases block the use of VoIP over 3G [16]. Another issue is that Quality-of-Service (QoS) is not guaranteed and that a SIP client on the mobile phone has no assurance that voice will actually work. Using the WiFi connection is another possibility where the data rate is usually higher and the level of quality is better but it still faces the same issues as the 3G connection.

Chapter 4

Mobile Operating Systems and Techniques

Running a contact center client on a mobile phone is very different from the standard desktop client. The desktop environments only purpose is to support the contact center agent and has a dedicated phone and screen. The mobile phone on the other hand has many more roles than just running the mobile agent client and can be interrupted by events such as incoming calls or SMS. The basic requirements of the mobile agent is that the client should keep the connection with the server alive during voice interactions. The client should also be able to maintain focus during the call so that the agent can read and update user data during a voice interaction.

In the following sections different mobile phone operating systems and techniques will be discussed.

4.1 Iphone

The Apple Iphone is a GSM phone using EDGE (2.5G) to increase download speeds. The phone model is currently not released on the Nordic market but a 3G version is planned for release during the later half of 2008. OSX is the operating system running on Apples desktop computers and has been modified to fit the smaller mobile device. The native programs are developed using a version of C called Objective-C and while no native 3rd party software is allowed to be installed on the device, webapps which package an AJAX web page into an application of sorts is allowed. In the upcoming 3G version of the Iphone the firmware is upgraded to version 2.0 where native programs are allowed. The application distribution will be done using an appstore provided by Apple. At the time of this report it is unknown if it is possible to sell programs to a specific user or if it will be available to everyone.

4.2 Symbian

Symbian OS is an operating system used in many Nokia and Sony Ericsson smartphones. The Symbian operating system has roots back to devices in the 1990s and has been created with the mindset that resources are very scarce. The native language of the Symbian OS is a non standard C++ implementation which has a steep learning curve

which requires the use of special techniques such as descriptors and the cleanup stack. This can cause relative simple programs to become harder to implement than in other environments. Many Symbian phones also support different versions of Java and script languages such as Perl and Python. The Nokia version of the Symbian OS comes with Python support and APIs for the different phone capabilities.

Starting with the release of Symbian 9.x applications that do anything more than read/write files and access the Internet will need to be digitally signed using the Symbian Signed program [13]. The Symbian Signed process is available for single devices used during development but signing an application that is meant for release needs to fulfill a specific requirements set by Symbian such as low memory footprint and responding to external events such as SMS and calls.

The applications for Symbian are developed in Java or C++ and accessing the telephone functions is possible with the CTelephony API. CTelephony provides a simple interface to the phones telephony system. It is possible to retrieve information about the phone such as network information, call functionality and phone settings. Dialing answering and controlling voice calls is also possible.

The Symbian user interface is supplied by either UIQ3 or S60.

4.2.1 UIQ3

UIQ3 is a graphic user interface on top of the Symbian OS that assists developers with increased tool support and ease multi platform development as well as defines the look and feel of the mobile device. This interface is mostly used by phones from SonyEricsson and Motorola.

4.2.2 Series60

Series60 (or S60) is the Nokia suite of standard applications and libraries for the Symbian OS. S60 is licenced by Nokia to other manufacturers such as Samsung and LG.

4.3 Windows Mobile

Windows Mobile is Microsofts operating system for the PDA and smartphone platforms. The biggest supplier of WM handsets is HTC. Nokia and SonyEricsson do not currently have any WM phones but the soon to be released X1 (which is produced by HTC) will feature WM 6.3.

The Windows Mobile SDK integrates tightly with the Microsoft Visual studio making it possible to create software in both native (Visual C++) and managed (Visual C-sharp, Visual Basic .NET) code. Starting with version 5.0 of Windows Mobile telephone behavior is exposed through the Telephone API. And while it is possible to make outgoing calls, handling incoming calls is not possible.

4.4 Android

Android is Googles open source operating system for mobile devices launched in November 2007. Members of the Google Open Handset Alliance such as Samsung and HTC are expected to ship mobile phones with the Android operating system in late 2008. The programs on the Android operating system run under a virtual machine called the

Dalvik Engine which is similar to a Java VM. Java code is converted to the Dalvik format and run by the Dalvik engine. By allowing programmers to code in Java and to run the programs in the Dalvik Engine, Google can use a lot of Java developers experience. The conversion of Java code to the Dalvik Engine also allows Google to bypass restrictions of the Sun Java license. As there are no real Android phones released yet it is uncertain if there will be legal problems for the Google Android project over the Java to Dalvik conversion. Another uncertainty is to what degree call management such as dialing and answering the phone will be allowed.

4.5 Java ME

Java ME (often referred to as J2ME) is Suns Java technology adapted to fit a resource-constrained device such as mobile phones. The Java ME technology is based on three elements. First a configuration that provides the most basic set of libraries and virtual machine capabilities for a broad range of devices. Then a profile which is a set of APIs that support a narrower range of devices and lastly an optional package of technology-specific APIs.

Over time the Java ME platform has been divided into two base configurations, one to fit small mobile devices and one to be targeted towards more capable mobile devices like smart-phones and set-top boxes. The configuration for small devices is referred to as the Connected Limited Device Configuration (CLDC) and the more capable configuration is referred to as Connected Device Profile (CDC). A widely adopted example is to combine the CLDC with the Mobile Information Device Profile (MIDP) in order to provide a complete Java application environment for mobile phones and other devices with similar capabilities. The CDC profile provides a much more conventional Java runtime environment compared to the CLDC profile. The developer support for the CDC profile has not been as strong as for the CLDC profile which has led SonyEricsson to drop support for CDC in its future models.

This modularity and scalability are defined by J2ME technology in a model with three layers of software built upon the Host Operating System of the device. These layers are illustrated in figure 4.1

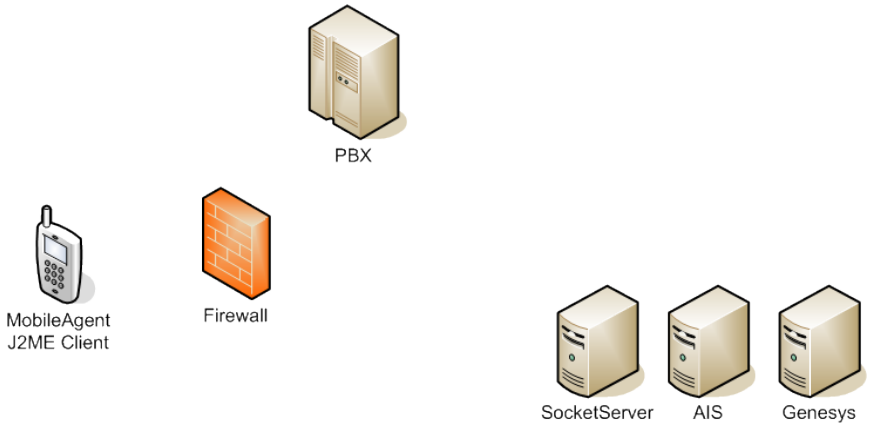

Figure 4.1: J2ME configurations and profiles

4.5.1 CLDC and CDC Java Configurations

Connected Limited Device Configuration CLDC and Connected Device Configuration CDC are two java profiles specifically tailored for mobile devices. The CLDC profile is used on most phones today and the CDC profile is used on more high end business phones. Since CLDC is much more popular and supported in comparison with the CDC profile the client for the mobile agent should should preferably be developed using this profile [4][3].

4.5.2 MIDP

The Mobile Information Device Profile (MIDP) is a specification for the use of Java on embedded devices such as mobile phones and PDAs. MIDP is a part of the Java ME platform and sits on top of the Connected Limited Device Configuration (CLDC). CLDC and MIDP provide the core application functionality required by mobile applications with a rich set of Java APIs [5].

MIDP 1.0

The first version of MIDP was aimed at what is now very low end mobile phones and had a lot of limitations. No support for full screen mode, no audio and no socket based communication. Some of these limitations were avoided by the use of vendor-specific APIs but that reduced portability.

MIDP 2.0

The limitations of the 1.0 specification were addressed in the MIDP 2.0 specification. A new security model where applications can be signed and granted access to previously restricted APIs were added. Other new features is the ability to have full screen Java applications and new APIs for games and multimedia. Most mobile phones on the market supporting the Java ME standard uses MIDP 2.0 [1].

MIDP 3.0

The next evolution of MIDP is the MIDP 3.0 standard which currently is still in the planning stages. At the time of this report the current specification does not inlcude call management [11]. Some of the new features in this version are:

- Specify proper firewalling, runtime behaviors, and lifecycle management issues for MIDlets
- Enable auto-launched MIDlets (e.g. started at platform boot time)
- Enable background MIDlets (e.g. UI-less)
- Increased functionality in all areas (UI, games, network)
- IPV6

4.5.3 Over-The-Air Provisioning

Over-The-Air provisioning means installation or upgrade using a network connection on the mobile phone. This feature can help the distribution the mobile agent software by sending out a SMS or email containing the URL of the location of the software. By collecting data from the website providing the OTA service it is also possible to create a customized version using settings entered by the user [6].

4.5.4 Obfuscation

Obfuscating means to make obscure or unclear but it has other properties than to make the code harder to read. Obfuscating the code helps improve memory footprint, performance and to some degree help protect the intellectual property. Long declarative function names such as SendMessageToInputHandler get renamed to A. This can reduce the size and memory requirements of a program by more than 50 percent.

4.5.5 Java Community Process

Java Community Process (JCP) is the community process that develop and revise the Java technology specifications. They suggest a Java Specification Request, JSR, on how java should be extended. Phones with Java technology support a collection of these JSRs such as Bluetooth (JSR-82) and Web Service (JSR-172). The Mobile Telephony API version 2 (JSR-304) is currently in early draft mode and will hopefully be implemented in future phones and will enable call management for Java ME.

4.5.6 GUI libraries

J4ME is an open source library for developing graphic user interfaces for J2ME. It provides a unified look and feel across all J2ME devices. The name J4ME is a play on words and means Java For ME (ME standing for both Java ME and "'me"') and is not to be confused with J2ME. Using the standard way of displaying graphics on a Java ME device you get different implementations of how a graphical component such as a text box should work, how a check box works and so on. It uses the GameCanvas to provide full screen support. It handles both keys and touch screen as well as buttons and text input is unified across platforms. The J4ME project uses the Apache 2.0 license which allows for both open and closed source projects to use the J4ME library.

An alternative to the J4ME libraries is the J2ME Polish project which is a suite of tools aimed at mobile developers. One of the tools is the UI toolkit called Lush which allows the UI to be described outside of the applications source code and helps create a unified look across different mobile platforms. The applications developed using J2ME Polish must either be open source or pay a licensing fees.

4.5.7 Web Services

Web services is a common way of making web applications exchange data with its clients. SOAP is the protocol used for exchanging the XML-based messages over the network using HTTP/HTTPS for transport. Using XML as the basis for the messages have many advantages. The messages are text based and self-documenting make it easy to read and understand. It is suited to represent common data structures and is platform independent. The main concern with using XML is that the XML representation is much

larger than a binary representation which may be an issue in a mobile environment. The Java ME Web service APIs is limited compared to the full blown Java APIs. The Java ME devices are typically limited by the amount of web services they can support. Some of those limitations are listed below.

- No support for service endpoints (no J2ME device based web services only clients).
- No service discovery support (UDDI).
- Validating parsers are not required given memory and processor needs.
- SOAP 1.1 encoding is not supported.
- SOAP messages with attachments are not supported.
- SOAP message handlers are not supported.
- XSLT is not supported.
- DOM is not supported.
- No support for dynamic proxies or dynamic invocation interface (DII).

4.6 Genesys Expert Contact

In the Genesys system there is a feature called Genesys Expert Contact that basically is an expert connected to the contact center. Unlike an agent who handles hundreds of calls each day, an expert only handles those calls where his unique knowledge is needed. The expert is treated specially by the call routing software in that it sends out a request to the expert asking if the expert would like to take this call or not. This is different to a normal agent which is not asked to take an interaction, the interaction is just sent to an available agent. This expert is not expected to be at the contact center but can be at any location. This fits well with a prototype software in that it is expected that an expert will have a bigger benefit from using a mobile agent client [15].

4.7 Web based AJAX Mobile Client

Asynchronous JavaScript and XML (AJAX) is a technique for creating more dynamic web pages. It is possible to update a page without the need for a complete page reload. This makes it possible to create responsive applications on the web. The AIM Web client developed by TeliaSonera uses AJAX technologies to access the Genesys contact center functionality. Gaining access to these features from the default browser installed on all new phones would make for a very short development cycle and very portable. The client software would not need to be installed on the agents mobile and updates can be made on the application server without the need to upgrade each phone.

An inventory of the mobile phones browsers reveal that there is little or no support for AJAX. Several developers have announced that they will include AJAX support in their next version. Seeing how protected the phone capabilities are even while using the APIs, call control will most probably not be allowed for an AJAX web site. Silverlight and Flash are both popular browser extensions on the desktop both promising mobile versions and make for an interesting client when the mobile browsers market matures.

Browser	Version	AJAX	Comments
Opera Mobile	9.5	Yes	Not released yet
Opera Mini	4.1	No	Uses an Opera proxy for browsing
IE Mobile	6.1	Yes	No phone control
Firefox Mobile	X.x	Yes	No specifications yet but promises full desktop like browsing

Table 4.1: Browser Support For AJAX

Chapter 5

Network Integration

An alternative or complement to developing a client for a mobile device is to integrate with services in the mobile network.

5.1 CallGuide and Virtual Call Center

CallGuide is a contact center product developed at TeliaSonera that comes in a version called Virtual Call Center (VCC), where the call traffic control is handled by servers in the network instead of a typical PBX. This concept of having the traffic controlled by servers in the network could be extended to mobile phones. Using this concept opens up possibilities to gain access to extra information about the mobile phones, such as monitoring presence, call status, in and out of reach and location. The call control is handled by a Parlay gateway which will be replaced by another technology soon, possibly an integration with IMS and SIP. The benefits of having call control in the network is that no servers are needed at the customer since everything is handled by the operator.

5.2 IP Multimedia Subsystem IMS

IMS is a multimedia platform for mobile networks such as UMTS, EDGE, GPRS and WLAN. This multimedia platform is by many leading companies considered the next evolutionary step of the standard mobile network (also standard telephony and broadband services). The IMS platform offers a variety of basic network services, such as session control services (including registration, routing and roaming) and secure authentication. In addition to these features a more flexible charging and payment system is possible. One of the problems with the current network implementation is that Quality of Service (QoS) can not be guarenteed for IP services on a mobile device. IMS enables operators to mix and match equipment and applications from multiple vendors and enables mobile users to access their personal set of services wherever they roam. Standard way of interoperability access awareness policy support charging, security and QoS.

A good example illustrating some of the possibilities of IMS is that a user can seamlessly add video to an ongoing voice session. It is also possible to transfer this video session to the TV when the user gets home. Another example is using different

access networks from the same terminal and change between a 3G connection to WiFi when the user enters a WLAN hotspot.

To help developers create the nescessary programs for the IMS platform Ericsson has developed the IMS Client Platform (ICP). The ICP frameworks goal is to help the deployment of IMS capable clients. The ICP framework is available as an extension for Java ME (JSR-281) which is currently under development [18].

Using IMS for integration with a mobile agent is a promising prospect. One example could be signing in using a special contact center profile. Incoming interactions would come up as a interaction request on the screen.

Chapter 6

Security

Accessing the contact center from a mobile device has a much larger need of security features such as encryption and how sensitive data should be handled compared to a standard client running inside a secure company network. For instance the user data sent from the contact center application to an agent can contain sensitive information such as pin codes or account numbers. An internal network can reasonable be thought of as secure but a mobile agent is not located on the local network but outside the secure network where the traffic can be hijacked. The mobile agent client can connect using either a 3G UMTS connection or a public WLAN. Using a public network has many security issues i.e. an unencrypted or a low security WLAN can easily be wiretapped and a WLAN using WEP for encryption can be cracked in as little as three minutes [20]. Companies with a high demands on security such as banks may also consider attacks on the 3G network link to the mobile agent [sec3g08]. Another set of security issues that arrises due to the nature of the device itself is much more likely to be lost or stolen. Sensitive data should never be permanently stored on the device.

The client authentication with the server can be made more secure with a digest authentication instead of a plain text authentication. The servers reply is concatenated with the users password and then the resulting string is hashed SHA-1 algorithm [21]. This will ensure that the users password is safe. Another way is to encrypt the entire socket to the server using Transport Layer Security (TLS see section 6.5). A company may have a very complicated security policy with DMZs, firewalls and proxies. Also firewalls and proxies on the mobile client network can be difficult to get a clear view of and it is most likely not possible to change. The client should use a minimum of ports to simplify the work needed to connect a mobile agent with the contact center.

6.1 Java ME Security Policy

Java includes the concept of sandbox for mobile code where the rest of the system is isolated from the untrusted program. A MIDlet whis is a Java program for embedded devices, can be executed inside the sandbox where it does not have access to privileged system resources. With digital signatures and signed code it is possible to verify the integrity and the signer of the application and thus trust the code. A MIDlet suite that needs access to a protected API must request the corresponding permissions. Recording sound, taking a picture and connecting to the Internet all require the user to grant these permissions at runtime. Both HTTPS and SSL is built in the MIDP 2.0 standard and

can be used to encrypt the session with the contact center server [7].

6.2 Windows Mobile

Most Windows Mobile 6 Professional devices are configured with a one-tier access model:

- Drivers and pre-boot applications require privileged mode signing.
- Signed applications can call all APIs without user permission.
- Most Windows Mobile 6 Professional devices allow unsigned applications to call all APIs with a single user permission at initial load. However, mobile operators may ship future devices in a mode that requires signing for execution.

The requirement for application signing is set by the mobile operator and it is possible to install an application without signing it. The code signing is done using Mobile2Market which connects an application to a independent software vendor (ISV). The main advantage with this procedure is that an malicious software can have its certificate revoked. The process involves acquiring a certificate and having the software tested by an external test vendor. If Microsoft finds the submitted info and the results from the test vendor acceptable the program is signed and can now be distributed freely. Standard security protocols such as TLS and SSL are also included.

6.3 Symbian Security Policy

Symbian OS 9.x uses a capability based model which ensures that sensitive operations, such as, modifying user data, making calls and using network connections, can only be accessed by applications which have been certified by an appropriate signing authority thus making it traceable. This signing authority is provided via the Symbian Signed program to which a program can be submitted. The Symbian Signed program has many levels of certificates ranging from a certificate only valid on a single mobile phone to the use of an independent test house [12]. Additional platform security includes full encryption and certificate management as well as the standard security protocols (HTTPS, SSL, TLS).

6.4 Android Security Policy

The Android security model is based on the Linux 2.6 kernel and as such is a multi process system where each application runs in its own process space. The security between the application and the system is enforced at the process level through standard Linux facilities, such as user and group IDs that are assigned to the application. Another level of security is provided by a permission mechanism that enforces restrictions on specific operations such as sending SMS, making a phone call and accessing the phone book. A basic Android application has no permissions associated with it, meaning it can not use any protected features of the device. In order to access the protected features a manifest file listing the restricted features that the program is required to access. When the application is installed the requested permissions are granted by the package installer based on checks with trusted authorities and interaction with the user. No checks are done while the application is running. It is either granted during install or it will silently

fail without notifying the user. This is done to reduce the number of permission requests a typical Java ME program asks the user. A potential issue with this approach is that it can be hard for the user to understand the implications of granting a capability to an application. An example of this is granting access to a program for sending SMS and the program starts sending premium-rate SMS. The second issue with this security model is that there is no way to authenticate the origin of a program since there is no equivalent to the Symbian and Java signed application.

6.5 Transport Layer Security

Transport Layer Security (TLS) is based on the Secure Socket Layer (SSL) and is a widely adopted security protocol for web applications which is implemented below the application layer. When a client accesses a TLS resource, such as opening a HTTPS connection, a handshake is performed. In this handshake the user is authenticated and a symmetric key is generated. Public key encryption is used to submit the shared key along with other additional data. When the client and server are authenticated symmetric key encryption is used for all data sent between them. During the handshake the client and the server agrees on what encryption algorithm to use thus making TLS flexible. TLS solves a lot of the security requirements of a mobile agent but encrypting the data traffic will have an impact on battery life and server load [19].

The Bouncy Castle Crypto package is an open source project that provides a Java implementation of cryptographic algorithms including a lightweight TLS client that runs on Java ME devices and is a possible candidate for implementing cryptography on the mobile agent client [2]. Both the Symbian and the Android SDKs include a TLS client.

6.6 Firewalls and End User Configuration

A firewall is typically used by a company to limit outside access to resources behind the firewall. In a standard configuration the firewall is located on the border of the company network and access to computers on the inside is restricted. For services that needs to be accessed from the outside special exceptions can be added to the firewall configuration. The next step is to have a zone called a Demilitarized Zone (DMZ) with a set of rules somewhere between the internal network on one side and the Internet on the other. In a corporate environment these sets of rules and policies need a clear understanding of how the mobile agent is connected to the contact center framework. Using a single port and a defined set of IP addresses will help simplify the process of configuring the firewall for a mobile agent.

Chapter 7

The Environment

7.1 Target Device: SonyEricsson P1i

The Sony Ericsson P1i smart phone was selected as the target device because of the following properties: Widespread deployment in companies and the phone supports WLAN, 3G, GSM GPRS and both the CLDC and the CDC java profiles. The operating system of the P1i phone is Symbian so it is also possible to develop a native C++ program. Other useful features are the QWERTY keyboard which will help future interaction types such as email and chat. List of technical specifications:

Screen	2.6 inch 240x320
Operating System	Symbian UIQ 3
Memory	128MB RAM 256MB Flash Up to 160MB available for user Max JAR size: unlimited Max SIS size: unlimited
Java	CLDC 1.1 CDC 1.0 MIDP 2.0 WMA 1.1
Networks	UMTS 2100 MHz GSM 900/1800/1900 MHz
Battery	Talktime GSM 10h / UMTS 3.5h Standby GSM 440h / UMTS 350h

Table 7.1: SonyEricsson P1i Specifications

7.2 Utility Tools

7.2.1 Wireless Toolkit 2.x

The Wireless Toolkit (WTK) is Sun's set of tools to help assist software developers create and debug Java ME programs. Besides a set of emulators for different screen sizes the WTK has tools for building, packaging and signing MIDlet suites. Simulated OTA provisioning and creating obfuscated JAR files is also provided. Since there is no console screen on a mobile phone device debugging can be very demanding. Luckily with the WTK emulator it is possible to monitor network traffic, exceptions and other useful debug info that can greatly help debugging. The Wireless Toolkit does not include a source code editor but the toolkit can be integrated in other IDEs such as Eclipse.

7.2.2 SonyEricsson SDK

The SonyEricsson SDK is a modified version of the WTK aimed at better supporting SonyEricsson devices with a more accurate emulator and the ability to perform "'on device debugging"'. There is unfortunately no emulator for the P1i phone but a similar sized screen was provided by the JP8 emulator.

7.3 Integrated Development Environment

Both the WTK and the SE SDK provide the necessary tools for mobile Java development but an IDE that integrate them well has been of great value during the development of the prototype.

7.3.1 Eclipse with Java ME plugin

The Eclipse IDE environment provides a good source code editor and project management and is easily extended with the numerous plugins available. EclipseME is a plugin to help develop J2ME MIDLets from within the Eclipse IDE. The EclipseME plugin does the grunt work of connecting the Wireless Toolkits to the Eclipse development environment.

7.4 Testing Environment

In order to test the application the Java server proxy needs access to a AIS server install which in turn has access to the Genesys framework.

7.4.1 Agent Interaction Suite

The Agent Interaction Suite (AIS) is a Java server written at TeliaSonera CIS using the Genesys Interaction SDK to communicate with the contact center. The AIS server exposes web services to the Agent Interaction Manager Web (AIM Web) client which is a fully functional contact center agent. These web services are used by the Java server proxy.

7.4.2 Apache Tomcat

The AIS server runs on the Apache Tomcat J2EE servlet engine.

Chapter 8

Application Design

Deciding between the operating systems and programming languages with all their different possibilities and restrictions is not a trivial task. The SonyEricsson P1i was chosen as the prototype platform because of its strong connectivity options (UMTS, EDGE, WiFi) and has a very widespread deployment in the Nordic countries. The Symbian operating system can run both native C++ programs as well as Java programs (both CDC and CLDC). Since I had no previous experience coding C++ for Symbian the decision was made that the prototype should be built in Java ME with the CLDC profile. A proxy server should be placed between the mobile agent client and the Genesys platform to help reduce the network traffic sent to the mobile agent. Additional features such as keeping the mobile network connection open with the help of "'keep alive messages"', possible integration with the mobile network and to offer a clear entry path for firewall configuration and a clear distinction of what traffic will be sent through a firewall.

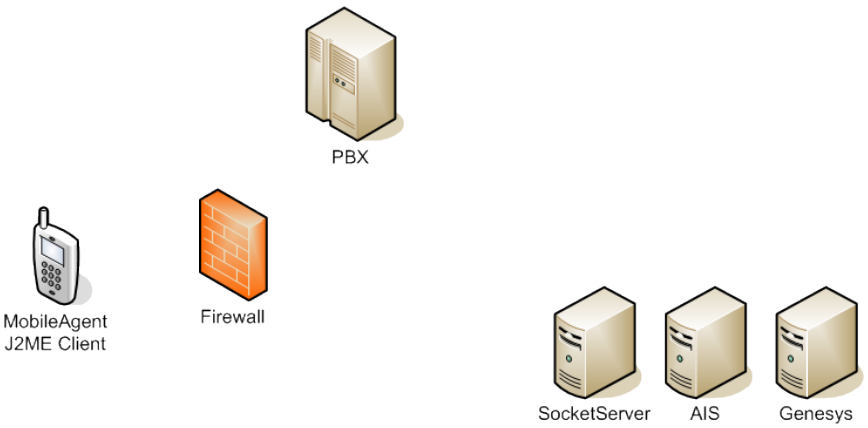

Figure 8.1: System Design

The proxy server is located between the Java ME application on the mobile phone and the AIS server. The AIS server is in turn connected to the Genesys servers. Using web services exposed by the AIS the Java proxy handles the mobile agents connection. This approach helps reduce the network traffic for the mobile agent. Another requirement is that the Java proxy should also handle phones disconnecting, changing cells and other

types of scenarios a typical mobile user will experience.

8.1 Java Server

The proxy server waits for connections from clients and when a new user connects, initiates a new SOAP connection to the AIS server. After the user has logged in the proxy waits for events either from the AIS server or commands issued by the user. When the contact center wants to transfer a call to the client a request is sent out to the client. The user can then choose if he wants to accept the call. This process of handling interactions is different from a standard contact center agent where a new interaction gets sent out to a free agent. The interaction handling is based on the Expert Contact feature in Genesys. After the user has accepted the interaction the call will be transferred out to the mobile phone, when the call has ended the agent changes his status to available and is now ready to handle more interactions. The data sent from the J2ME client can be a small and effective protocol based on the premises of sending data over a mobile network.

8.2 Java ME Client

The client software is a standard Java ME application and can run in the background while still maintain a connection to the proxy server. The GUI is run in full screen mode and uses the J4ME GUI libraries to allow capture of UI clicks and data input.

8.3 Error Handling

A mobile client is much more prone to disconnect than a standard desktop client because of the wireless network link. The server should try to gracefully handle the most common causes for when the client unintentionally disconnects. If the client is disconnected due to bad reception, the server side should try to disconnect the agent from the contact center. The client should detect that is has lost connection and try to automatically connect again.

When a mobile phone connects to the internet it usually goes through the operators proxy server. In the Telia 3G network a connection socket is closed after 30 minutes of inactivity so in order to prevent a mobile agent client from disconnecting a keep-alive message should be sent at regular intervals. If the user receives an incoming phone call while using the mobile agent the client software should change the users status to not ready.

Chapter 9

Implementation

Both the client on the mobile phone and the proxy server were implemented in Java and the client runs in the Java ME MIDP 2.0 environment. The proxy server runs under a normal Java VM and uses web services to connect to the Genesys framework using the AIS server front end. Using the network connection on the phone (3G or WLAN) the client connects to the proxy server which in turns uses web services to connect to the contact center. After the initial authentication and login requests the client is ready for incoming interactions.

9.1 Agent Interaction Suite

Using the Agent Interaction Layer (AIL) from Genesys TeliaSonera has developed the Agent Interaction Suite (AIS) which consists of an Agent Interaction Server and a client called Agent Interaction Manager Web (AIM Web). The purpose of this server is to provide an interface for the AIM Web client using web services but this interface is also available for other programs. This is the point to which the Java server proxy connects to the Genesys framework.

9.2 Web Services

Having the client talk directly with the AIS server using a web service interface has the advantage of not having to translate each SOAP messages sent from the AIS to a proprietary binary format sent to the client as well as eliminates the need of a proxy between the AIS servers and the client.

By using a server proxy between the client and the AIS server the issues caused by SOAP could be eliminated. The overhead caused by the XML messages are only sent between the proxy server and the AIS server. By using another protocol for sending messages between the client and the proxy server data traffic costs can be kept at a minimum. Another issue addressed by the proxy server model is that having the AIS web service interface exposed to the Internet may cause unwanted security issues.

9.3 WSDL2Java

WSDL2Java is a tool that can automatically create the web service stubs from a Web Service Description Language (WSDL) file. The WSDL file containing a description of how the Java functions work was supplied by the AIS server. Instead of manually creating the SOAP messages that need to be sent it is now possible to just call the remote method like any other java method. This simplifies the interface to the AIS web services.

9.4 Socket Server

The server software, based on the web service stubs generated by the WSDL2Java application, listens on port 7580 for client connections. When a connection to this socket is made a new thread handling that client is created and the server begins to listen for the client commands. When a login command is received from the client the server creates a new SOAP connection to the AIS server. Every client command is then detected and the appropriate AIS SOAP command is called. Incoming commands from the AIS server are processed and sent to the client.

9.5 Mobile Agent Client

The Mobile Agent client was developed using the J4ME libraries to easily capture touch screen clicks and handle text input. When the Mobile Agent application has started a login screen is displayed. Here the user can change his username, password and server to connect to. When the user is happy with the settings and clicks on "'OK"' the message handler thread starts up and sends a login request to the Socket Server and waits for a login OK message. When that message is received the thread enters the main message handling loop and waits for input for either the Socket Server or user input. The user can enter commands by clicking the message button on the screen or if the device does not have touch screen by clicking the corresponding key.

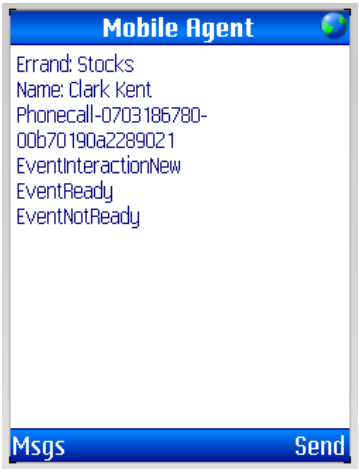

Figure 9.1: The Mobile Agent client

Chapter 10

Results

The development phase had many challenging problems. The first major problem was that in order to test the client on a real device a port needed to be opened in the firewall to the local development environment. Fortunately the emulator proved to be an sufficient test and later when the client was tested on a real device it worked.

10.1 The Client

The mobile agent client was developed in Java ME using CLDC and MIDP and the entire process of packaging and deploying the client was done by EclipseME with no problems. Due to the client being developed before the server a server emulator was needed. The issue was solved using a program called netcat which can open a listening socket. With this socket open it was possible to connect from the mobile agent and emulate the server replies. Having a socket open on the local machine caused issues when the client software ran on a real device. Access to the development machine from the outside mobile network was blocked by a firewall. Later in the development phase when it was possible to test the server from a real device it worked as expected thanks to the emulator being accurate enough for the mobile agent prototype. Only one minor issue was found with the emulator in that the J4ME GUI library had some small graphic problem (borders where not aligned properly). During the tests of the mobile agent client it was discovered that a connection was closed after 30 minutes of no traffic du to the configuration of proxies in the Telia mobile network.

10.2 The Server

Generating the stubs for the server was done using the AIS WSDL file and the remote calls worked without a single issue. Using just one socket for both incoming and outgoing messages proved to be difficult a socket read would block the iteration loop. This was solved using a function in Java that checks if the socket has incoming bytes.

Chapter 11

Conclusions

Finding out what platform is the most suited for a mobile agent platform presented itself to be a rather challenging process. One of the first goals decided on was to try to make the mobile agent as light-weight and mobile as possible. This ruled out any sort of laptop/UMPC solution using a VPN connection to the contact center. Instead focus was aimed at finding out how a mobile phone could be used as a mobile agent.

There are essentially three ways of developing applications for a mobile phone: Java, Symbian C++ and Windows Mobile. The Android platform from Google has not yet been released and the Apple Iphone does not currently allow 3rd party applications and while the introduction of Iphone 2.0 does include plans for support of 3rd party applications the model only allows pre approved applications to be sold and installed from Itunes, which is unacceptable for a business application such as this.

The SonyEricsson P1i used in this thesis as the prototype device supports both native Symbian programs as well as Java ME (both the CDC and CLDC profiles). The Symbian SDK gave me the impression that even the smallest project has a very long start up time. For instance just creating a simple application requires you to setup a complicated environment with a complicated set of tools and certificates.

After Java ME was selected as the intended platform for the prototype I discovered that even though calling a number from Java is supported, answering the phone is not. When the phone starts ringing the call application takes over focus from the mobile agent client and the user has to find the program again. Using the Expert Contact feature of Genesys the user is essentially asked if he wants to take this call. The call is then transferred out to the mobile agent and answered like a normal call. When the call is finished the user has to switch back to the Java program running in the background and send a message back to the server notifying it that the user is now available for calls again. Also the Java program is sent to the background when an incoming call is received and if the user wants to update or view user data while in a call he must manually switch back to the client. With the introduction of IMS some of these shortcommings could be avoided by signing in using a special contact center profile. Incoming interactions request would pop up in the default user interface. Quality of Service introduced with IMS also allows for a client using VoIP for voice.

Security is a very important part of a mobile agent platform because of the security threats introduced by having the agent outside of the contact center. Encrypting the link between the mobile phone and the contact center is a good start but perhaps certain user data should never be sent to a mobile agent and be filtered earlier due to

security reasons. Having signed applications helps the user determine that the program downloaded really is the correct program and has not been altered. The downside of the increased security on a mobile device is that the process of signing a program can be very costly and complicated.

Unfortunately the enhanced security models on a mobile phone introduces many restrictions compared to a normal desktop computer. Restrictions on how many aspects of the phone a program is allowed to control. For other programs just needing access to the internet or the camera this is not a huge problem but for a mobile agent needing access to answering and dialing and maintaining focus during incoming phone calls this poses a problem.

Another issue is that different operators have different settings for how a mobile phone connects to the internet. Some operators even block access to the internet and only allow web access through the built-in browser. Using port 80 which is the default port for web traffic can also cause problems since the protocol between client and the server proxy is not HTML and can be rejected or cached by a HTTP proxy between the client and the server.

The standard mobile browsers do not yet provide AJAX capabilities (Internet Explorer on Windows Mobile being the only exception). But both Android and Opera have stated AJAX support on there next line of mobile browsers. Browser plugins such as Adobe Flash and Microsoft Silverlight which enable much more control of how a user can interact with the web could be future possibilities of implementing a mobile agent.

11.1 Restrictions

Due to time constraints the connection from the client to the proxy and from the proxy to the AIS server is not encrypted. Also the connection made by the phone to the proxy times out after 30 minutes if there is no traffic. This is because the proxy residing in the TeliaSonera mobile network closes unused connections and the keep-alive feature is not implemented. Also the only interaction type handled is voice.

11.2 Future work

Due to the time constraints the client sends the password in clear text and a authentication scheme should be implemented. This can be improved with a basic digest authentication in a first step and then later be extended to use encryption with TLS.

Another issue is that the server proxy does not send keep alive messages and the client connection will drop after a certain time (30 minutes in the TeliaSonera network) if there is no network activity.

The server does not have any logic for handling a client disconnect and since the server is still connected via web services to the AIS server and Genesys it is possible that interaction requests will be sent to a client.

Opera which are providing the default browser for Symbian is working on a new version of the Opera Mobile browser which is said to have support for AJAX. This is very interesting since it would be possible to have the client inside a standard browser. Things like client upgrades and firewall issues would become trivial. It is however unlikely that advanced features such as call control will be supported from the AJAX

enabled browser. The concept of the expert contact where the mobile agent accepts an incoming phone call request would work even with the possible restrictions.

Only voice interactions are currently supported, adding more interaction types like email and chat or even video are definitely on the future work todo list. When IMS is fully deployed in the mobile and fixed networks maybe just signing in with the contact center profile will be all that is necessary to login and all calls will be connected using SIP and IMS.

More and more applications move out to the web and the user can access them from anywhere using only a standard browser. The mobile browsers are not currently up to the task of handling an application such as the mobile client, which needs access to the phones capabilities. Increased AJAX support has been announced for the next generation browsers and the gap up to the desktop versions will decrease. Browser plugins such as Silverlight and Flash will help enable much more control of how a user can interact with a web site.

Chapter 12

Acknowledgements

I would like to thank my two supervisors at TeliaSonera Ali Nouri and Jonas Flygare for their help and support. Carl-Magnus Ekermann for his input on network integration, Björn Eriksson for his report on CallGuide Mobile Agent and a big thank you to the entire PSI team at CIS Uppsala.

References

[1] Otto Kolsi, Teemupekka Virtanen, *MIDP 2.0 Security Enhancements*. Proceedings of the 37th Hawaii International Conference on System Sciences, 2004.

[2] Jason Weiss, *Java Cryptography Extensions: Practical Guide for Programmers*. Morgan Kaufmann 1 edition, 2004.

[3] Sun Microsystems, *CLDC HotSpot Implementation*. Sun Microsystems, 2005. [Online]. Available: http://java.sun.com/j2me/docs/pdf/CLDC-HI_whitepaper-February_2005.pdf

[4] Sun Microsystems, *Micro Edition Connected Device Configuration*. Sun Microsystems, 2005. [Online]. Available: http://java.sun.com/j2me/docs/j2me_cdc.pdf

[5] Sun Microsystems, *Mobile Information Device Profile*. Sun Microsystems, 2005. [Online]. Available: http://java.sun.com/products/midp/midp-ds.pdf

[6] Sun Microsystems, *Over The Air User Initiated Provisioning Recommended Practice*. Sun Microsystems, 2001. [Online]. Available: http://java.sun.com/products/midp/OTAProvisioning-1.0.pdf

[7] Cynthia Bloch, Annette Wagner, *MIDP 2.0 Style Guide for the Java 2 Platform, Micro Edition*. Addison-Wesley Professional, 2003.

[8] JSR-118, *Mobile Information Device Profile 2.0*. JSR-118. [Online]. Available: http://jcp.org/en/jsr/detail?id=118

[9] JSR-253, *Mobile Telephoni API*. JSR-253. [Online]. Available: http://jcp.org/en/jsr/detail?id=253

[10] JSR-304, *Mobile Telephoni API version 2*. JSR-304. [Online]. Available: http://jcp.org/en/jsr/detail?id=304

[11] JSR-271, *Mobile Information Device Profile 3*. JSR-271. [Online]. Available: http://jcp.org/en/jsr/detail?id=271

[12] Ben Morris, *Platform Security and Symbian Signed: Foundation for a Secure Platform*. Symbian Developer Network, 2008. [Online]. Available: http://developer.symbian.com/main/downloads/papers/PlatSec_and_Symbian_Signed.pdf

[13] Symbian, *Symbian Signed*. Symbian, 2008. [Online]. Available: https://www.symbiansigned.com

[14] Google, *What is Android*. Google, 2008. [Online]. Available: http://code.google.com/android/what-is-android.html

[15] Genesys, *Genesys Expert Contact 7.2* Genesys, 2007.

[16] BBC News, *T-Mobile must open Truphone lines*. BBC News, 2007. [Online]. Available: http://news.bbc.co.uk/1/hi/business/6901945.stm

[17] Ericsson, *Introduction to IMS*. Ericsson Whitepaper 284 23-8123 Uen Rev A, 2007.

[18] JSR-281, *IMS Services API*. JSR-281, 2008. [Online]. Available: http://jcp.org/en/jsr/detail?id=281

[19] Dierks, T. and C. Allen, *The TLS Protocol Version 1.0*. RFC 2246, 1999. [Online]. Available: http://www.ietf.org/rfc/rfc2246.txt

[20] E. Tews, R-P. Weinmann, A. Pyshkin, *Breaking 104 bit WEP in less than 60 seconds*. 2007. [Online]. Available: http://eprint.iacr.org/2007/120.pdf

[21] National Institute of Standards and Technology, *Secure Hash Standard*. 1995. Federal Information Processing Standards Publication 180-1. [Online]. Available: http://www.itl.nist.gov/fipspubs/fip180-1.htm

[22] C. Andersson, *GPRS and 3G Wireless Applications*. Wiley Computer Publishing, 2004.

Appendix A

Glossary

PBX Private Branch eXchange.

VoIP Voice over IP.

VPN Virtual Private Network. Used to create a secure tunnel over a public network.

AJAX Asynchronous JavaScript and XML. A technique for updating a web page without refreshing the entire page.

SOAP SOAP is a protocol for exchanging XML-based messages over computer networks.

GPRS General Packet Radio Service. Packet oriented mobile data service.