Device Discovery in Device Management Systems for Cellular Networks

BARTLOMIEJ SPIZEWSKI

KTH Information and Communication Technology

Master of Science Thesis
Stockholm, Sweden 2007

COS/CCS 2007-16

Device Discovery in Device Management Systems for Cellular Networks

Master of Science Thesis

Bartlomiej Spizewski

Examiner:
Professor Gerald Maguire

Supervisor:
Yipu Gao

Stockholm, Sweden
22 March 2007

Abstract

As mobile phones get increasingly complicated the demands for an effective firmware update service increase. A proposed solution is Firmware Over The Air (FOTA) and the Open Mobile Alliance's Device Management where mobile phones can be updated and managed via the mobile phone network. However, before these operations can be carried out, all FOTA capable mobile phones that should be served must be discovered and registered with a distributor of updates. The information provided must be sufficient to uniquely identify devices, initiate a Device Management session, and determine if a firmware update is needed. This thesis addresses the problems that a solution in automatically collecting this information. Several solutions are presented and their suitability evaluated on the basis of defined and analyzed requirements. The solutions most thoroughly examined are various manual solutions, retrieval of information from core network nodes, and utilizing the Short Message Service (SMS) or Unstructured Supplementary Service Data (USSD).

A phone application has been implemented according to the requirements from the Chinese network operator China Mobile Communications Corporation (CMCC). It is a part of a solution in which the information is delivered via a SMS session. The design and development phase of the application is described, accompanied by a brief description of the Symbian OS and the working environment (tools, devices, etc.) needed to implement this solution. This work took place at the Sony Ericsson office in Beijing, China.

The application implemented is robust and it is impossible to avoid registration, furthermore the user can not be exposed to acknowledgement messages. It has been made possible on the cost of decreased phone performance (a few kB of memory) since the application runs all the time. Malfunctioning phone or network may hinder registration.

Sammanfattning

Mobiltelefoner blir ständigt mer komplicerade vilket medför att efterfrågan av en effektiv lösning för uppdateringar av mjukvaran i mobiltelefonerna ökar. Lösningen är Firmware Over The Air (FOTA) och Device Management; mobiltelefonerna uppdateras och sköts via mobiltelefonnätverket. Men innan förfaranden kan exekveras måste alla mobiltelefoner med FOTA som ska omfattas av tjänsten upptäckas och registreras hos den som distribuerar uppdateringarna. Den information som måste levereras måste vara tillräcklig för att kunna identifiera mobiltelefonen, genomföra en Device Management session och avgöra om en uppdatering av mjukvaran är nödvändig. Detta examensarbete behandlar de problem som en lösning i vilken information tillhandahålls möter. Ett flertal lösningar presenteras och deras lämplighet utvärderas på basis av definierade och analyserade krav. De mest ingående undersökta lösningarna är olika manuella lösningar, insamling av information från noderna av kärnnätverket samt utnyttjande av SMS eller USSD.

En telefonapplikation har implementeras enligt krav från den kinesiska operatören CMCC. Applikationen är en del av en lösning i vilken informationen levereras via en SMS-session. Applikationens design och utvecklingsfasen är beskriven, samt en översiktlig beskrivning av Symbian operativsystem och utvecklingsmiljön (verktyg, mobiltelefoner, etc.) som behövdes för att implementera lösningen. Detta arbete genomfördes på Sony Ericssons kontor i Beijing, Kina.

Preface

I would like to start by thanking Xiaoguang “Ziteng” Chen at Sony Ericsson in Beijing who answered all my questions and helped me through the difficult times of the development phase. Numerous are those who supported me during my stay in Beijing and made it a very pleasant one. In particular, I owe a gratitude to Yipu Gao, my supervisor, who guided me when I was off course. I am very grateful for the opportunity given to me by Sony Ericsson in Beijing to write my master thesis at their office and learn to know the Chinese culture. Mats Lundgren, Ib Green, Amos You, and Vanessa Zhou made all this possible.

I would also like to specially thank Professor Gerald “Chip” Maguire, my examiner at the Royal Institute of Technology, who guided me and provided invaluable comments on my work.

Bartlomiej Spizewski, Stockholm, Sweden, March 2007

Table of contents

1 Introduction

1.1 Background

As with all new innovations, we don’t understand how we could have lived without them. Instead of physically going to the bank we can pay our bills on the internet, we can communicate via e-mail more effectively compared to earlier means of communication, and by using MP3-files we can have our whole collection of music on a mobile phone and listen to whatever we want. This master thesis concerns the implementation of one of the next in line of these innovations, namely Firmware Over The Air (FOTA). FOTA enables remote firmware updates via the wireless network. The devices that will be updated by using FOTA in this context are mobile phones and PDAs. The number of FOTA-enabled phones was expected to have reached 150 million units by year-end 2006 [1]. As mobile phones have become more and more sophisticated and the amount of code has correspondingly increased, the number of significant software errors has increased, hence the need for an effective update service is evident. Also as security issues for mobile phones have become more important [2] the number of updates is expected to increase. Although users now can perform firmware updates via their home PCs [3] the FOTA solution is potentially better since the user only needs to trigger a search for an update via a menu on their phone or by answering an update request initiated by the vendor or operator. The service is always available and available worldwide (assuming that you are located in an area where your operator has either coverage or a roaming agreement) [4], but the cost of receiving the update may vary depending upon where you are and who your operator has agreements with. So instead of running to a service center on your lunch break you can update your phone during the commercial break when watching a sports match on TV.

In order to provide an effective service, the FOTA provider needs to know the identity of each of the devices which it should provide updates for, how to initiate a DM session to that device, and how to determine if a firmware update is needed. During the first stage of this thesis project a phone application was designed and developed, as part of a solution to this problem. This implementation was designed to meet China Mobile Communications Corporation (CMCC) requirements, see Appendix A. This work took place at the Sony Ericsson office in Beijing, China. The next phase of this thesis was to examine and evaluate other solutions to the problem and analyze their suitability. The CMCC solution was also included in the evaluation. The reason why the design and implementation took place before the comparison with other solutions was because CMCC already had chosen a solution that Sony Ericsson needed to implement as soon as possible.

At first glance, the CMCC, on behalf of the Chinese government, may seem having some hidden registering and surveillance agenda. However, that is not the case since it can simply be required as it has been in the USA and several other countries [39].

1.2 Problem description

In order to provide an effective service, the FOTA provider needs to know the IMEI, manufacturer, model, software version, and MSISDN of every FOTA enabled mobile phone that is to be served. Registration should not be a cost for the user, in terms of money, time, or effort. Additionally neither the mobile phone's, nor the network's, performance should be significantly reduced, and preferably no changes to implementations or new implementations in the nodes of the core network or radio access network should be necessary.

1.3 Scope and Assumptions

The scope of this thesis is to analyze and gather all information needed to implement a self-registration application on a Sony Ericsson smartphone. The requirements of the application are defined in Appendix A. The implementation was made on a Sony Ericsson smartphone, model M608 or W958, running Symbian OS version 9.1.

The scope of this thesis includes mobile phone network architectures, Device Management systems, user experiences, and business aspects, since solutions other than that used for the CMCC self-registration are described and evaluated.

The registration solution needed to work in a 2.5G mobile phone network, specifically a GSM and GPRS network, and preferably also in China's upcoming 3G network, using TD-SCDMA. This assumption was made to focus the project and because the required implementation was meant for the Chinese market. When specific information about TD-SCDMA was unavailable another 3G technology was used for reference. The assumption was that the choice of 3G technologies to refer to will not have a major impact upon the proposed solution, since the solution was already expected to work in both GSM and GPRS networks.

1.4 Outline of thesis

This report begins with a chapter describing the thesis parts of the technologies underlying Device Management, mobile phone networks, and mobile phones. Additionally, the actors and their roles in a Device Management system are analyzed and described (chapter 2). In the next chapter the requirements forming the basis by which solutions are evaluated are defined and analyzed (chapter 3). The solutions are presented, discussed and analyzed in the following chapter. A discussion about distributing the functionality via SIM-cards is also included in the chapter (chapter 4). Examples of approaches to the problem implemented by companies are presented in the *Related work* chapter (chapter 5). The implementation and design, accompanied by a brief description of the Symbian OS, Symbian C++ language, and the working environment are described in the CMCC self-registration chapter (chapter 6). The thesis ends with a conclusions (chapter 7) chapter and a future work chapter where an attempt to foretell the future of DM registration is made (chapter 8).

2 Device management, mobile phone networks and mobile phones

In order to analyze solutions, and to understand the need for self-registration, knowledge is needed of the relevant elements of the technology underlying Device Management (DM), mobile phone networks, and mobile phones. The "big picture" is also needed in order to put this work into an appropriate context; this is described in section 2.4.

2.1 Mobile phone networks

In a mobile phone network, a device, in most cases a mobile phone, communicates via base stations which cover large areas. Each base station is responsible for handling the devices within its coverage area. An area served by an antenna in a channel defines a cell, mobile phone networks hence are often referred to as cellular networks. The base stations are the last hop link of a mobile phone network. These base stations are connected into a system which connects the mobile phone to the PSTN, internet, or other mobile phones. Most mobile phone networks are today built on 2G [5] or 3G [6] technologies. The self-registration solution needed to be implemented so that it would work in CMCC's 2G network, which uses the GSM standard [8], and preferably in the emerging Chinese 3G network built on the TD-SCDMA standard. GSM combined with GPRS is often described as 2.5G [7].

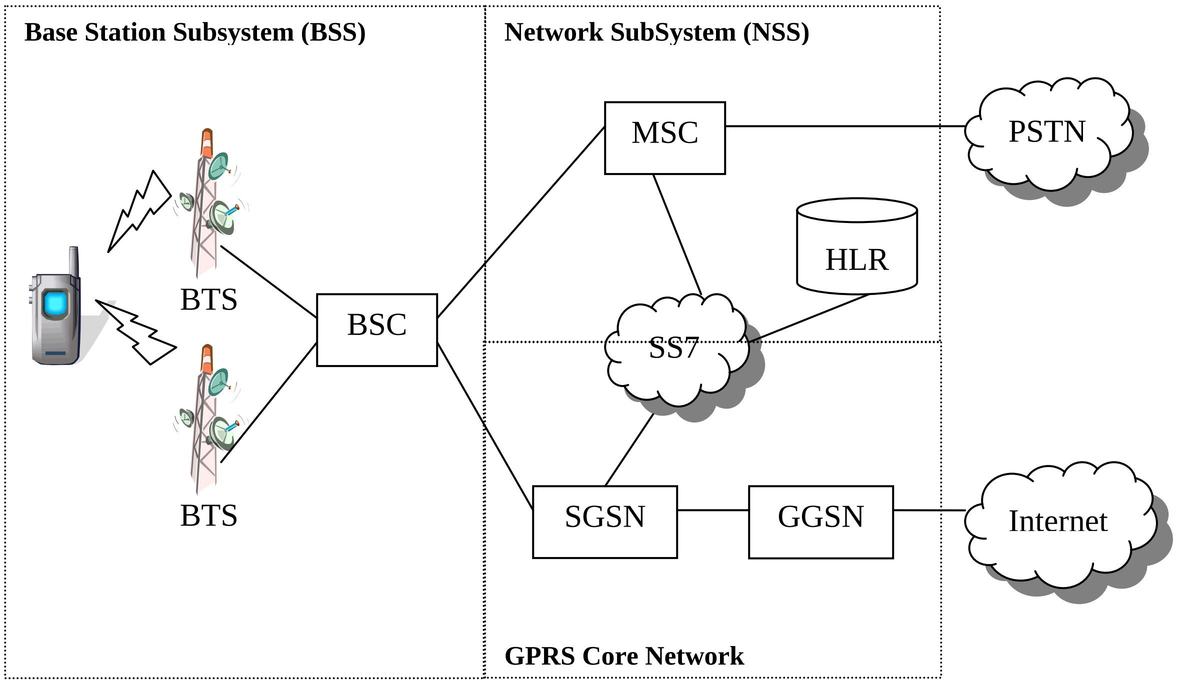


Figure 1. GSM network

As figure 1 shows, the structure of a GSM network can be divided into three sections: the Base Station Subsystem (BSS), the Network and Switching Subsystem (NSS), and the GPRS Core Network. The BSS includes all the base station coverage areas, as described above. In GSM networks the BSS includes Base Transceiver Stations (BTSs) and Base Station Controllers (BSC). The BTS transmits and receives radio signals, while the BSC controls one or more BTSs. [59]

The NSS provides the switching intelligence which manages the communication between mobile phones; mobile phones and the PSTN; and mobile phones and the Internet (or other packet networks). Other GSM services such as SMS are implemented here. The main component of the NSS is the Mobile Switching Centre (MSC) which actually implements the services mentioned above. A MSC communicates with BSCs and is responsible for all mobile phones roaming within the its area serviced by all of the attached BTSs. Mobility management, i.e. handling phones moving between areas, is a key feature of the MSC. The MSC also collects billing information. A MSC communicates with other MSCs forming a larger network. A database called the Home Location Register (HLR) is connected to the MSC. The HLR is a central database containing information about the operator’s Subscriber Identity Modules, i.e. SIM-cards. These modules provide the information to identify and authenticate subscribers who are authorized to utilize this operator’s network and its services. Each SIM contains at least one International Mobile Subscriber Identity (IMSI) which is a unique number that is used to identify this subscription in the HLR. Additional information stored in the HLR includes the Mobile Station International Subscriber Directory Number (MSISDN), this is the “phone” number used when making or receiving regular calls, making data calls, or fax calls. The MSISDNs associated with the SIM are used as primary keys when a call is made to a subscriber. The HLR also hold records of which GSM services a user has access to, if the user is allowed to use GPRS, and the network current location of the user. A network operator maintains records only for its own subscribers in the HLR. In case of roaming users, i.e. users from another operator for with whom this operator has roaming agreement, the user’s network location is temporary recorded in a Visitor Location Register (VLR). The VLR provides information about mobile phones that have attached to this operator’s MSC. When a user is roaming, the VLR communicates with neighboring VLRs and MSCs and maintains records, including the Temporary Mobile Subscriber Identity (TMSI). Both roaming users and users in their home network utilize this temporary IMSI, rather than the IMSI to increase subscriber confidentiality. Ideally, the actually IMSI is sent as rarely as possible because when it is sent it is sent the clear, hence it is very easy to determine that a given subscribe is present within this cell.

The GPRS Core Network is integrated with the rest of the network, but was added as an extension to the original GSM network providing packet switched services to the rest of the network. The GPRS Core Network communicates with the NSS, but implements its own mobility management, session management, and charging functionality [21]. The Serving GPRS Support Node (SGSN) has much the same functions as the MSC in the NSS, i.e. authorization and authentication, mobility and session management, charging, etc.; but is concerned exclusively with packet-switched data. It is connected to BSCs and interacts with the MSC and HLR/VLR. The Gateway GPRS Support Node acts as a router by routing IP packets to the correct SGSN or to the Internet. The SGSN utilizes mobility management to support the mobile phone as it moves between cells.

The Mobile Application Part (MAP) is an application layer GSM protocol used by various nodes in the NSS when they communicate with each other. MAP is implemented on top of the Signaling System #7 (SS7) family of protocols and the main operations executed by MAP are: mobility management, operation and maintenance, call handling, supplementary services, and SMS [30].

2.1.1 GPRS

General Packet Radio Service (GPRS) is an extension to GSM networks and it enables packet switched communications for mobile phones. Today most GSM networks support GPRS. Traditional phone calls are circuit switched sessions (implementing a virtual connection), meaning that a whole circuit, or channel, is reserved for each call. Packet switching allows faster downloads of data since it utilizes the network capacity better and allows higher degrees of statistical multiplexing because of the bursty nature of much of the data transmissions. For voice transfer to support traditional phone calls, circuit switching is still considered the best solution, although UMTS is moving to eliminate all circuit switching within the network infrastructure and circuit switching will only exist on the radio links themselves. A GPRS capable mobile phone connects automatically to the GPRS, if the network operator allows it. Most mobile phones are logically connected all the time to GPRS, but it is only a logical network connection providing information to the user that GPRS is available and providing information to the GPRS network about the phone. To actually send user data via the GPRS network a GPRS session must first be initiated by the user. Network activation of a GPRS session is generally not supported [7]. Today most GPRS capable devices can utilize both GPRS and GSM services (such as voice calls or SMS), but often only one can be used at a time (although it is possible in many networks to send SMS traffic via the GPRS network). When for example a voice call is made, the GPRS session is suspended and resumed after the call has ended.

2.1.2 Circuit switched data connections

Circuit switched data connections have been available in GSM networks since the beginning of GSM. They are preferred when a high quality of service is needed since a fixed amount of bandwidth is reserved. Data transfers are made by emulating dial-up functionality where the phone acts as a modem. However the data is not actually transferred on top of a voice call, but rather the frames are marked as containing circuit switched data, hence the data does not have to be modulated, thus the phone is not actually acting as a modem (even though the devices acts as if it were a modem).

2.1.3 SMS

Short Message Service (SMS) is a GSM service which makes it possible for mobile phones to send and receive short text messages. SMS is often used when a mobile phone user needs to interact with automated systems, e.g. when ordering services. The payload, size of a SMS message, is limited to140 bytes. The message can be encoded using 7, 8, or 16 bit characters. Larger messages can be sent by concatenating multiple, i.e. the larger message is segmented/divided up into smaller messages. SMS messages can be sent to specific users or to all users in a specified cell. From the mobile phone, the message is sent to a Short Message Service Centre (SMSC) which provides a store and forward system for SMS messages. SMSCs utilize a store and forward mechanism by which messages can be resent to users that are not currently reachable. There is no guarantee that a message will be delivered; although a user can request delivery reports notifications of failed deliveries these are considered unreliable [9]. Binary content such as ring tones or configuration and programming data can also be sent in SMS messages, if extensions to the GSM SMS specification are utilized. Unfortunately SMS uses the signaling channel, i.e. the channel used to control services. Thus the effective data rate is low. Additionally, the use of the control channel by SMS traffic can adversely affect other services, although a message can be received during a voice call. SMS has been transplanted to TD-SCDMA, but the GSM specification differs from the 3G specification and the implementation at the application level is different [46]. Today many operators send SMS via GPRS as it avoids the negative effects upon other uses of the control channel.

2.1.4 USSD

Unstructured Supplementary Service Data (USSD) is a technology in GSM networks similar to SMS in that it sends data utilizing the signaling channel. However, USSD provides session-based connections unlike, SMS which offers only a store and forward message transaction oriented solution. Because of this the response times of USSD are shorter, up to 7 times faster than SMS for two way transactions [29]. Network operators often make use of USSD for users to query the network and receive a fast response. The user simply types in a combination of decimal digits, asterisks (*) and hash-symbols (#) on their mobile phone. For example, when using a pre-paid card, the user can do an account balance inquiry by typing in *111#, send, and the balance is displayed on the display. Most GSM mobile phones today support USSD Phase 2 where a dialogue can be established and several operations sent; rather than a single request and response as was the case with USSD Phase 1 [31]. An USSD server is connected to the user’s HLR via MAP and to the server providing the service via TCP/IP [29]. The payload of a USSD message is 182 characters, i.e. slightly more than SMS. There are two modes of USSD: the Man Machine Interface (MMI) mode as described above and the application mode which is intended to be used by applications in the network and their peer applications on a mobile phone [28]. The network applications can be located in the VLR, HLR, MSC, or some server on the internet [28], i.e. a mobile phone can communicate with these nodes by using USSD. The IMSI used to send the USSD message is passed along with the message. USSD is always supported by the subscriber's home network - even if the subscriber is currently roaming.

A billing mechanism for USSD is in most cases not implemented. However, there are some rare cases where network operators done so [32], and the charging is then based on the duration of the session. USSD is a capability implemented in all GSM phones [10] and has been incorporated in the TD-SCDMA standard [11].

2.1.5 WAP Push

Wireless Application Protocol (WAP) is a standard which enables applications on mobile phones to access some internet services. WAP Push messages are SMS or USSD messages containing a link to a WAP address which the user can choose to open. A WAP address is similar to a web URL. Using WAP Push messages a user can be informed about and subsequently choose to download content from a WAP server. The user’s phone number can be supplied in the session. WAP Push has major uses in distribution of mobile phone content, such as applications, from suppliers to users. However, WAP’s poor security and other failings lead to a major reworking of the WAP protocol stack, which today looks much like the standard TCP/IP stack.

2.2 Mobile phones

The most relevant information about mobile phones in conjunction with this thesis is the IMEI/IMEISV and the contents of the SIM. The SIM is formally a separate entity according to the GSM specifications, but the functionalities of the mobile phone and the SIM are so tightly connected that they can logically be considered a single entity. Detailed features of the mobile phones that concern the implementation part of this thesis project are described in chapter 6.

2.2.1 IMEI and IMEISV

The International Mobile Equipment Identity (IMEI) is a number unique to every cellular device. It is used for identification of the specific mobile device when it attaches to the cellular network. The network operator can prevent stolen phones from making use of the network if the IMEI of the phone is on a ban-list. However, emergency calls can still be made with a stolen phone [22], in addition, emergency calls can be made with a phone without a SIM card. The IMEI consists of decimal digits only and is composed of the Type Allocation Code (TAC), the Serial Number (SNR), and a check digit, totaling 15 digits. The TAC is issued by an international GSM standards body and indicates the country of origin and uniquely identifies the model of the phone. The following 6 digits are the serial number that together with the TAC identifies a specific device. The last digit, the check digit, is never transmitted; its purpose is to prevent manual data entry transmission errors, i.e. check if an IMEI typed in by a person is correct.

The IMEISV is the same as the IMEI except that a two digit Software Version Number (SVN) is attached at the end. Mobile phone manufacturers can choose to implement the SVN, and it must be sent along with the IMEI, if the IMEI is requested by the network. The check digit does not consider the SVN.

According to the 3GPP specifications [22] a network operator can make administrative use of the IMEI by defining three registers: white list, black list, and grey list. The white list contains equipment identities that are allowed to use the network. The black list is composed of all equipment identities that shouldn’t be allowed to use services of the network. A grey list can also be used for listing equipment that is to be “tracked by the network (for evaluation or other purposes)”.

2.2.2 The Subscriber Identity Module

The Subscriber Identity Module (SIM) is the removable smart card containing identification and authentication data. This information is transmitted when a user inserts the SIM into the mobile phone and powers the phone on. A SIM is internationally identified by its International Circuit Card ID (ICCID); it is stored in the SIM and printed or engraved on the SIM. The IMSI is stored in the SIM and is used by the network operators to identify a subscriber, as described in section 2.1. The composition of the IMSI is described in detail below. An authentication key and information about the local area is also in the SIM, along with a number for the SMSC, and the service provider’s name and service dialing numbers.

The SIM Application Toolkit is a GSM standard which provides mechanisms allowing applications on the SIM to interact with the mobile phone. The SIM can in this way be programmed to perform operations such as sending SMS or USSD messages [12].

2.2.2.1 International Mobile Subscriber Identifier

The International Mobile Subscriber Identity (IMSI) is composed of a Mobile Country Code (MCC), a Mobile Network Code (MNC), and a unique subscriber number (MSIN), totaling 15 digits. The MCC is three digits long and consists of an ITU-T country code. The MNC contains two or three digits and is used in combination with the MCC to uniquely identify the network operator.

2.3 Device Management

Device Management (DM) in the context of this thesis is the technology as defined by the Open Mobile Alliance (OMA) [15]: "Device Management includes, but is not restricted to setting initial configuration information in Devices, subsequent updates of persistent information in devices, retrieval of management information from devices, and processing events and alarms generated by devices."

OMA is an industry forum for developing market driven, interoperable mobile service enablers [19]. Its most important role is to provide technical specifications. The DM technology described and investigated in this thesis is based on OMA's DM standard [44].

The main idea of DM is that the user shouldn't be forced to go to one of a few physical locations, such as a retailer, to get the latest update. Instead, the mobile phone, or some other mobile device, can be upgraded or managed anywhere an internet connection is available. DM data can be transferred to the phone via various technologies such as data cables, Bluetooth, or IR data transmission [14], but the real advantage is that data can be transferred through a mobile phone network. Protocols that can be used are WAP, HTTP, or OBEX. A binary representation of the Extensible Markup Language (XML), WAP Binary XML (WBXML), is used to exchange data at the application level. It is compact, as string and tag tables are used, hence suitable for narrow band networks such as the 2G mobile phone network. WBXML carries the same information as XML, thus no conversion to XML is needed. Although WBXML was originally meant for wireless networks, it is of course not limited to that [45]. A client-server network architecture is used where the mobile devices are the clients and are managed by the servers. Both the server and the client are stateful. For security, SSL/TSL is often used to protect the session between the client and server. A session can either be initiated by the client, by e.g. choosing an "update" menu item on the phone, or the server, by sending a WAP Push message.

The OMA DM implementation in a device consists of the OMA DM Protocol and the OMA DM User Agent. The OMA DM Protocol controls establishment, management, and termination of the DM session. The OMA DM User Agent executes commands and generates status messages. All data is organized in a management tree, the DM-tree. The DM-tree contains a root node along with interior and leaf nodes. The leaf nodes hold the data, which can be a single value or a file. The root and interior nodes are used to structure the data and to address the data with a URI. [33]

To ensure interoperability, OMA often specifies management objects. The DM-tree is populated when the device is provisioned, i.e. when a device is made usable with a network. The Firmware Update Management Object (FUMO) is defined in the OMA specification for firmware updating over the air and is the key part of the OMA DM protocol on which this thesis focuses.

2.3.1 Bootstrapping and registration

Bootstrapping is the process in which a DM enabled device is provisioned. The first step is to configure a management session initiated by the user. Basically the address of the DM-server needs to be stored so that the device knows where to get further information when it is first powered on. A device can either be bootstrapped before it reaches the user, factory bootstrapped, or if the server is informed about the device at some time afterwards. In both cases the personalized device must be registered in the server before a management session can be started by the server, this is where registration comes into the picture. When factory bootstrapped, the DM provider must know the phone number/network address of the phone before a DM session can be initiated by the server. When not factory bootstrapped, the phone number/network address must be registered so the phone can be bootstrapped and so that the server can later initiate DM sessions. The device does not necessary need to be provisioned in the factory, for example the device can be provisioned by personnel at a sales outlet. While provisioning at these locations might be even preferred since the data may be fresher, this does not change the basic scenario. The mobile phones that are not factory bootstrapped (for example sold outside an operator's normal sales line, e.g. second-hand) are often the most important to register since their configuration is very likely wrong since another operator might have provisioned the mobile phone, and the phone may need to be reconfigured for this new operator.

2.4 The Device Management System

The purpose of this chapter is to introduce how the Device Management system works; along with its actors and their roles. It is important to have a clear picture of this since it can have a major impact on the DM registration solution. The actors are: the manufacturer, the seller, the DM provider, the network operator, and of course the user. The actors are not necessarily from different companies, but they all need to cooperate for the system to work.

2.4.1 Manufacturer

The manufacturer is the one that provides the physical mobile phones and software updates. This includes companies such as Sony Ericsson, Nokia, or Motorola, but could also be a company, such as Flextronics or Cellon, that have developed a mobile phone which other companies sells under their own brand. These later companies are referred to as an Original Equipment Manufacturer (OEM) or Original Design Manufacturer (ODM), depending on how large their role has been in the design [35]. The manufacturer may buy a DM provider's services, if the manufacturer is concerned that the network operator won't provide these services to the user.

2.4.2 Sellers

The seller sells mobile phones and possibly accompanying subscriptions to users. Sellers can be a chain, such as The Phone House in Sweden, that usually, but not necessarily, sells mobile phones with a subscription. The seller can also be a network operator selling phones with subscriptions directly in their shops or on-line. The manufacturer might also be the seller when mobile phones are sold to employees or provided as a benefit. While network operators are the biggest customers of the manufacturers, the manufacturers also sell phones to other organizations. When the manufacturer sells phones directly to users a subscription is seldom included. Additionally there is also the case when a phone is sold between private persons, i.e. second hand, where subscriptions are very rarely included in the deal. The best DM registration solution should cover all these cases.

2.4.3 DM providers

The DM provider is often a company that has developed a solution for Device Management and provides it to a network operator or manufacturer. A DM provider could sell the service directly to users, only utilizing an operator’s network, but this has thus far not been the approach since the network operator often has an established (and close) relationship both to their subscribers (users) and one or more manufacturers. When a network operator chooses their DM provider, the DM provider should preferably present a ready solution, i.e. a DM server and a solution for DM registration which should be as universal as possible so it can be sold to other network operators. A DM provider’s services can be bought directly by a manufacturer or another organization. DM providers include companies such as Bitfone, Synchronica, and Insignia.

2.4.4 Network operators

The network operators are also very keen on having DM system implemented since in those cases when mobile phones are sold with subscriptions, both the subscribers (users) and sellers prefer that the operator will offer FOTA. In case the seller is a chain, they will promote such an operator since they profit from not needing to provide the update service in their shops. Since mobile phones are getting increasingly complex the demand from sellers for FOTA technology is increasing. Today operators must adapt to their new role as software update distributors since they operate the network and provide service(s) to their subscribers. In the terms of customer care, FOTA is important because of this scenario: Two users have different operators, but the same phone model. One of the operators provides FOTA to all its subscribers, while the other does not. The user using the operator who does not provide FOTA updates experiences malfunctions because the phone is not updated while the other one does not. The first user may draw the conclusion that the malfunction is the operator’s fault, in a way he or she has right to do that, thus they may change operator. However, as the operator doesn’t want to be blamed for such failures and they can’t depend on the phone manufacturer actually making a product that works, they have to put time, money, and effort into correcting this increasingly common problem. Additionally, network upgrades can cause interoperability problems, which quickly need to be remedied via an effective update service, i.e. FOTA. A user may also feel better taken care of and remain faithful to this operator if the user changes to an operator and the new one offer updates (if they are suitable).

2.4.5 Users

The users need not to be private persons; the users could also be a company or other organization that buys mobile phones. However, the user is the one that everyone wants to keep happy and satisfied, since in the end, it is the user (subscriber) that pays. This generally means that the user should experience as few problems as possible.

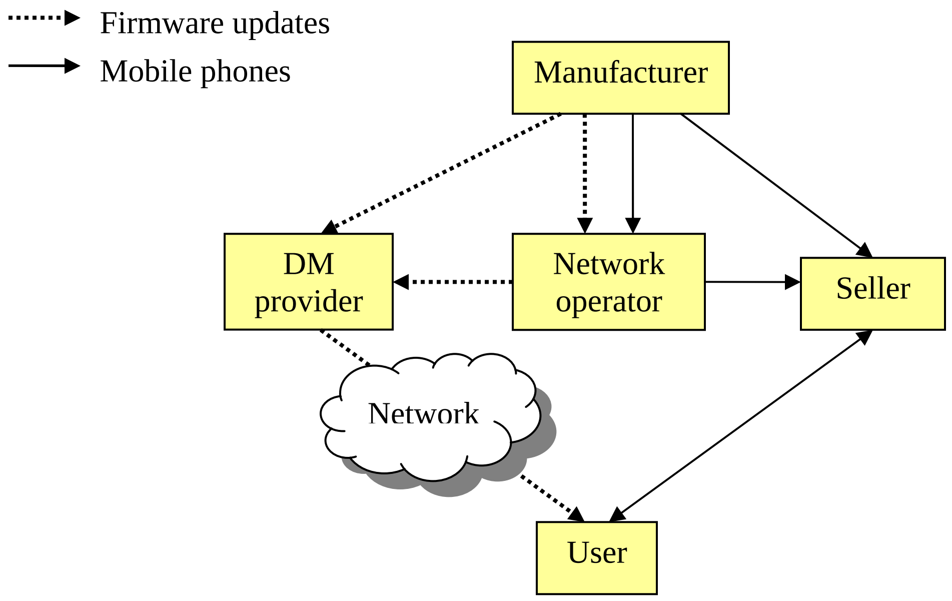


Figure 2. System actors.

The double-edge in the arrow between user and seller is meant to represent mobile phones sold second-hand. As mentioned before, the actors do not necessary need to work for different companies, e.g. the manufacturer is also a seller if phones are sold directly to users and a user becomes a seller when he sells a phone to other users.

3 Analysis and definition of DM registration requirements

The purpose of analyzing and defining DM registration requirements is to have a base to refer to when considering for DM registration solutions and to compare solutions and determine their appropriateness. The requirements are primarily based on *CMCC Self-registration requirements* (see appendix A).

The key words MUST, MUST NOT, MAY, SHOULD, and SHOULD NOT are to be interpreted as described in RFC 2119 [34].

3.1 DM requirements

1. All valid Firmware Update Management Object (FUMO) enabled mobile phones using a specific network operator’s network and a valid SIM issued by the operator MUST be registered in the DM-server selected by this operator.
2. The information received by the DM-server during the registration process MUST be sufficient to uniquely identify devices, initiate a DM session, and to determine if a firmware update is needed.
3. The registration MUST be carried out when a mobile phone is personalized, i.e. when a user starts using a SIM or when the SIM is changed.
4. The information sent to the DM-server SHOULD match the information saved in the DM-tree of the mobile phone.
5. The registration MUST NOT incur a cost to the user.
6. The user MAY be informed about the registration through the Graphical User Interface (GUI).
7. The registration SHOULD NOT reduce the mobile phone’s performance.
8. The registration SHOULD NOT reduce the network’s performance.
9. The registration SHOULD NOT imply implementations or require changes in the network operator’s core network.

3.2 Analysis of requirements

This section describes what the requirements stated above mean in practice.

Requirement 1

This means that under all circumstances, irrespective of the seller, the necessary information must be transferred to the relevant DM-server when the user starts using the mobile phone in this operator's network.

Requirement 2

The IMEI is ideal for identification and the MSISDN is needed to start a DM-session since it identifies the subscriber and is used to contact the device. The manufacturer, model, and software version is necessary to determine if a firmware update is needed.

Requirement 3

The entry in the DM-server is not useful if the DM-server can not contact the device. Hence the SIM should trigger the registration since it knows the MSISDN and it can know the address of the DM-server for this operator (as this data can be pre-stored in the SIM). Additionally, the DM provider and operator can be surer that the SIM sold together with the phone is going to be used with this phone, which they can’t be when a user leaves a shop with a purchased phone and SIM.

Requirement 4

A DM-session will encounter errors if the information does not match. The information may be different for various reasons, e.g. Sony Ericsson is sometimes SonyEricsson, and manufacturers may use different numbering of their software versions for the same versions (see network solution). It can be remedied if the DM provider obtains mapping information, but it is an unnecessary obstacle.

Requirement 5

Here cost includes monetary expense, time, or effort. This is necessary since users may be dissatisfied if registration costs them, particularly as they may see no return on this expense.

Requirement 6

There is no reason to explicitly inform the user that he or she has been registered since most people don’t like to be registered anywhere, especially without their permission. Informing the user that non-secret data has been sent makes little sense, and will result in unnecessary calls to support centers. However, it has no impact on the main functionality, and such notification can be done if regulations require it.

Requirement 7

It is unavoidable and acceptable that a mobile phone’s performance a decrease somewhat if an application is implemented that needs to transmit data. However, if this is only for a short time period and primary functionalities such as voice call aren’t affected too much, then the negative effects should be limited.

Requirement 8

The registration shouldn’t result in negative impact upon the operator’s network capacity, thus an insignificant increase in data traffic is acceptable.

Requirement 9

Network operators prefer that no applications are implemented in nodes of their core network, especially by companies serving other network operators. This requirement is applicable if Device Management is provided by such a third party, which is the most common case today [36] [37] [38]. Since DM providers probably don’t have access to the network’s core nodes during the development phase, and the solution should be universal, they need a solution that doesn’t require access to these nodes. This is especially true if the DM provider wants handset manufacturers as customers, but is not themselves an infrastructure manufacturer or operator. The purely technical suitability of a solution shouldn’t be judged by this requirement; it is more a business requirement from the DM provider’s point of view.

4 Solutions to meet the DM registration requirements

The chapter presents all possible solutions and evaluates them. The *Related work* chapter (chapter 5) should be read in conjunction with this one since it covers variants of the solutions. The relevant solutions are compared in a simple table on the basis of the requirements that were defined and analyzed in the previous chapter. The pros and cons of distributing the functionality are discussed in the last section.

4.1 Manual solutions

The manual registration solutions are the solutions where the registration is made manually, i.e. persons explicitly register their phone number on a web-site or the seller informs the DM provider that he has sold a phone with a specific SIM/MSISDN. Both solutions have several major flaws. In the first, there is no way to guarantee that all users register. Even if people are informed about the benefits, by costly campaigns, they can neglect or refuse to register. The risk is especially high when phones are sold between private persons. The risk that the IMEI sent is incorrect is very small since it is very easy to find, usually printed under the battery or retrieved using some simple command [25], and the check digit at the end of the IMEI minimizes the manual transcription error. The software version and model are usually easily retrievable, although phone models can have different names even if they look the same. However, it costs the user time and effort and the costs of web-sites or call centers must be considered. The solution where the vendor submits the information has the same flaws although they might not be so severe if the professionalism of the staff is high. They can use bar code readers to read the bar codes, usually printed under the battery, to retrieve correct information efficiently. Most shops already read the bar code off the box when they sell it as they need to account for the item being sold which is no longer in their inventory. But administrative costs rise and this approach does not provide a solution for phones sold second hand. The DM provider would need to provide a solution that integrates the functionality in the databases of all the sellers, or implement an application in all shops that connects direct to the DM provider’s server. This solution could be combined with the first one to remedy the second hand problem, but then it would still inherit all the flaws of this approach.

Closely related to this solution is the Ready to Run solution as described in the *Related work* chapter (chapter 5). The user is offered a chance to register through a GUI and the information is sent via SMS, hence the manual transcription error disappears. Since it has basically the same flaws as the manual solutions, it is considered as manual when comparing with other solutions.

A possibility that should be mentioned in this context is that the necessary information can be retrieved when users start using the synchronizing functionality, i.e. synchronize their calendar or contacts with a data synchronization server. The users would then register indirectly. However, this can only be considered as an add-on to the manual solutions since rather few users use this functionality. Additionally, the company providing synchronization would need to be the same as the one providing FOTA updates.

4.2 Network solution

This solution is based on the fact that the DM provider really only needs to know the IMEI and IMSI of FOTA capable phones. When a mobile phone attaches to the network the IMEI and IMSI are sent in the core network signaling [20]. The manufacturer and model of the phone can be retrieved from the IMEI. If the SVN field is present it is sent to the network when the IMEI is requested. The SVN field is not mandatory [22], but the network operator could request that feature. A mechanism could then be implemented, probably in the HLR, which informs the DM server when an IMEI from a FOTA capable mobile phone appears. Network operators have access to the IMEI allocation information [24], so the operator only needs to forward the information and facilitate the implementation of the mechanism.

However, the SVN field consists of two digits and most software versions are longer than that [25]. One solution to this would be if the device manufacturers numbered their official releases with only two digits. The software version in the DM-tree and on the DM-server would be the SVN. It is unlikely that more than 99 updates are released during a mobile phone's lifetime (SVN value 99 is reserved for future use) but it is still a constraint that shouldn't be neglected. In those cases where the SVN is implemented, the official software versions are already mapped to the SVN. Another solution would be to initiate a DM-session and retrieve the software version that way, but then the user must be bothered with an extra request, which in worst case only checks the software version (a firmware update can be done directly if the software is found to be old), resulting in a query to the user, which that he or she can refuse to answer.

Another problem may be the information regarding the manufacturer in the IMEI, the code indicates the manufacturer, which not necessary is the same as the brand of the phone. Companies with famous mobile phone brands sometimes buy mobile phones from other companies, OEMs, or ODMs, validate them, and sell them under their own brand. The DM provider requires the manufacturer name that is stored in the DM-tree which then would not match with the one registered for the specific code in the IMEI, since companies buying phones from OEMs or ODMs might want their brand name there. The same model can also have several different TAC-codes [27]. This could be solved if the node would have been identified via the TAC-code (the DM provider would then have to deal with several versions of a rather cryptic code), or mapping information could be provided to the DM provider so that TAC-codes could be translated before being stored in the server. This is not a major issue, but still something that should be addressed.

This solution doesn't necessarily need to be implemented in the HLR, as only the logs are needed. There is little urgency as the data does not need to be received in real time. However, the amount of the data in the logs can be substantial, and the network operators might be unwilling to provide the logs if the solution is outsourced. Fortunately, only the relevant information needs to be extracted and sent to the DM provider, but that would mean a cost for the operator. An effective link via which the information is sent to the DM provider must also be set up.

The great advantage of this solution is that it does not result in increased traffic in the network, it does not have an impact on the performance of the mobile phone, it does not require any costs to the user, and it is based upon already present information. The transmission of the IMEI and IMSI is very reliable since it is part of the core network signaling, i.e. there is no chance that a mobile phone will be missed. However, with the obstacles mentioned above in mind, and since information (sent via an effective link) from a node of the core network is needed, an easier solution would seem be to obtain the correct information directly.

4.3 The SMS solution

The SMS solution is the solution that has been implemented during the implementation phase of this thesis, i.e. equivalent to CMCC Self-registration. The detailed requirements are available in appendix A. When a CMCC-SIM is used on CMCC's network for the first time in a FUMO phone the IMEI, IMSI, model, and software version that are stored in the DM-tree of the phone are sent to the DM-server. A specific address and SMS-port is used. The MSISDN is parsed from the header of the SMS message upon arrival. The DM-server stores the information if it is valid, and sends an acknowledgement SMS message using a specific sender-address and SMS-port, containing the IMEI and a digit indicating if the registration was successful. An application on the phone, the same that sent the registration SMS, listens for the message on the specific port, parses the message and if the address is correct and the digit indicates success, stores the IMSI is in non-volatile memory in the phone. In case the digit indicates failure, or no acknowledgement message is received, the IMSI is not stored in the phone's memory. On next startup the application checks if the IMSI stored match the one in use. If there is a mismatch the registration message is sent, in that way no unnecessary registration messages are sent. Implementation details are presented in (chapter 6.6). The SMS message is free of charge (this is based on the specific address to which the SMS is sent to by the phone) and the registration is hidden from the user, i.e. there is no notification through the GUI. A user may switch the same SIM between two mobile phones. No registration messages will be sent when the user switch the SIM between the mobile phones after the first successful registration. If only the MSISDN is used, the entry will be replaced and it will look like the user is using only one phone, when he really uses two (or more) phones. There is a risk that the DM-server may send software updates to the wrong phone, but it is better than not knowing that there are two phones used. A second reason is that the IMEI is sent back in the acknowledgement message to make sure that the correct phone has been registered.

The disadvantages with this solution are the fact that the SMS is rather unreliable [9], and the server listening for SMS messages may sometimes be down, which was experienced during the implementation phase on the Bitfone test-server. A user might also experience a lower quality of the voice call if a call is made during the sending of the message. The impact on network performance is, as described below, insignificant.

4.3.1 Impact on network traffic

There are 150 million FOTA phones available on the world market today [1]. According to a report [41] the estimated SMS-traffic in the Asia-Pacific region alone is expected to have reached 802.4 billions SMS messages in year 2007. If all those phones would send a registration SMS in that region it would mean an increase of traffic by 0.019%. Worst case scenario: 5 billion registration SMS's sent in year 2010 when the SMS traffic is expected to have reached 1,212.7 trillion. This would result in a SMS traffic increase of

0.41%, which should be within the margin of error of the numbers presented in the report. Since SMS provides rather large revenues [41], the network operators will most probably take the appropriate measures to meet the demands of this increasing traffic.

The USSD solution

The USSD solution could basically have the same requirements as the CMCC Self-registration, see appendix A. Instead of SMS messages, USSD messages could be sent and received, i.e. an USSD-session executed. USSD is much faster [29] and more efficient since it is a real session. An USSD-session can be executed during a voice call, although the call quality may be lowered [29]. Both USSD and SMS use the same signaling channels; however the network traffic is more affected when using USSD since it keeps resources assigned. Traditional use of USSD (the MMI mode) direct USSD messages to the HLR, which means increased traffic on the channels between the MSCs and the HLR. Additionally a USSD message can only be sent to the user's home network and the connection is maintained for the entire duration of the USSD session, hence USSD can be expensive in terms of the resources which are assigned by not utilized. The SMS messages are directed from the MSC to a SMSC. Whether a network can cope with that is rather hard to know, given the information available, but the network can most probably handle two or three extra USSD-sessions during a mobile phones lifetime. Since USSD application mode would have been used, an application in e.g. the MSC’s could route the messages to a registration server. That is quite an enterprise; the SMS solution can utilize the already available SMSC.

Possibly, the SMS solution was chosen by Bitfone (and implicitly CMCC) since it is a well-known technology, but that is merely a speculation.

4.4 GPRS and Circuit Switched Data

Since GPRS and Circuit Switched Data (CSD) both are bearer technologies they should be mentioned in this context. A solution that sends the information via CSD can be ruled out directly since no voice call can be made during its execution and it is ineffective. Using GPRS could be considered if there were guarantees that all users had GPRS activated with the correct configurations and the coverage area was equivalent to that of SMS, which is not the case since we are dealing with often new and un-provisioned devices. On the basis of these facts the technologies are not investigated further and were not compared to the other solutions.

4.5 Comparison table

The purpose of the table below is to visualize how the different solutions fulfill the requirements defined in chapter 3. This table alone should not be used to judge the solutions, it just gives a hint.

Table 1. Comparison table of solutions.

	Manual	Network	SMS	USSD
Req. 1		**Met**	**Met**	**Met**
Req. 2	**Met**	**Met**	**Met**	**Met**
Req. 3		**Met**	**Met**	**Met**
Req. 4	**Met**		**Met**	**Met**
Req. 5		**Met**	**Met**	**Met**
Req. 6	**Optional**	**Optional**	**Optional**	**Optional**
Req. 7	**Met**	**Met**		
Req. 8	**Met**	**Met**		
Req. 9	**Met**		**Met**	**Met**

As table 1 shows, all solutions except the manual approach the mandatory requirements. The USSD and SMS solutions may have negative impact on the performance of the mobile phone and network, but it is rather insignificant. The network solution would have been the most optimal one if the SVN field was present in all phones and less limited, thus it could be used in the DM-tree; the issues regarding TAC-codes didn't need to be addressed; and none of the nodes of the core network were involved. However, the most probable reason why this solution has not been chosen is that the SMS solution is easier to implement, both in business and technical terms. It is somewhat comparable to application developers for personal computers that count on the high performance of the computer to hide the flaws of the application, and most of the time they are right. There is no reason to choose this solution rather choosing one that is utilizing existing services and provides correct information directly, especially when a network can handle the extra traffic. The manual solution has several variants, e.g. in the Ready to Run solution the registration is triggered by the SIM on first power up, meaning that the third requirement is partly met. What the variants have in common is that there is no approach concerning how to cover all cases for how phones are obtained by users, i.e. all vendor cases. They all also mean some kind of cost for the user, but in return the user can feel that they have some control of the registration process.

4.6 SIM distribution

SIM distribution is not a direct solution as in the sense of those mentioned above, but rather an idea of how the self-registration functionality could be distributed to the mobile phones. One could argue that instead of implementing an application in all phones, the application could reside on the SIM. The application on the SIM could send a SMS or USSD messages with the necessary information. But if this solution is to work as well as the other solutions, then all the SIM's in circulation would need to be recalled and replaced. This is because the operator wants to know when a FOTA enabled mobile phone is on their network and the only way to be sure that all those phones send the self-registration message would be by replacing all SIMs. The replacement of SIMs could be phased out by for example offering a new SIM when a mobile phone is sold, but this would still not remedy the fact that a new owner can use an old SIM. It could possibly work in countries like Sweden where people change mobile phones frequently, if the buyer agrees on changing the SIM and the operator thinks that it is worth the extra workload, but in order to reach full functionality, after a while, all the remaining SIM's without the self-registration application must be replaced by forcing the user to change by inactivating the SIM. All SIMs don't need to be inactivated at the same time; this could also be done in phases. But even when considering this, requiring the self-registration on all mobile phones concerned is a much better solution.

SIM distribution could possibly be justified if e.g. CMCC missed requiring the self-registration in a lot of phones before they reached the market. But since most FOTA enabled mobile phones have been available on the market for only one year [13], and the CMCC self-registration application requirement is older than that, SIM distribution is not justified. The CMCC self-registration application has been implemented in all FOTA capable Sony Ericsson mobile phones intended for the Chinese market. The only FOTA capable phones so far have been the Operating System Embedded (OSE) phones. The CMCC self-registration application implemented during the first phase of this thesis is intended for yet unreleased FOTA capable Symbian phones.

4.7 Positive side-effects

Besides its main functionality, the registration has rather unexpected positive side-effects. It has been a problem for the manufacturers to prove that the guarantee time has run out on returned phones. This is because it is impossible to show for sure exactly when a mobile phone was first used. When the self-registration is implemented, the manufacturer can retrieve that information from the operator. Actually, it is proposed that the information should be sent to a Sony Ericsson server, from which the operator could retrieve information, rather than vice versa.

Another positive side-effect is that stolen phones can be found easier than before. In order to find a stolen phone, the IMEI is needed. People often throw away or can't find the box or documents on which the IMEI is printed, thus it is not possible to find and block the phone. However, people seldom forget their mobile phone number, and by checking that entry in the database the IMEI can be retrieved.

5 Related work

It has been relatively hard to find related work since FOTA is a quite new technology and some variant of the manual solution has been used by most so far. Swapcom writes on their website that their automatic device discovery is a unique feature [43]. Solutions equivalent to CMCC Self-registration seem to be embraced by other network operators like Orange, but that information was only obtained by the word of mouth.

5.1 Ready to Run

Sony Ericsson has a related solution known as Ready-to-Run (R2R). An application that starts the first time a mobile phone is powered on. The user is asked if he or she wishes to download configurations for GPRS, MMS, WAP, and E-mail. If the user agrees, a temporary Sony Ericsson GPRS account is used to download the configurations. The addresses to the operator’s servers from which the data is downloaded are now stored on the phone. There is also a choice for the user to register, i.e. send an SMS with the same information as required in the registration solutions. It is, as mentioned, optional, and the user pays for it. There is no acknowledgement message. The solution can therefore be considered as the manual solution with a small cost for the user in terms of time, effort, and money. The application runs only once at the first startup since the primary use of the information is for guarantee time disputes.

5.2 Swapcom

Swapcom is a French company that sells solutions for remote management of devices, i.e. they are a DM provider. They have some interesting solutions described on their website [42]. One of them is the MMS troubleshooting solution. When a user attempts, but fails to attach while trying to use MMS, the MSISDN and IMEI are sent to a Device Management center. The MSISDN is used as the database key to tell if an account already has been created for this subscriber. If this is the case, the IMEI is checked and the settings corrected so that the user can start using the service. If no account is registered, the device is automatically provisioned. This is a troubleshooting solution for MMS settings, but it is the same basic idea as registering users when they perform remote synchronization, as described in the manual solution, i.e. retrieve necessary information when a user is accessing or tries to access other services.

Swapcom also provides an automatic device discovery through which data about devices is gathered when a device attaches to the network. The details are rather scarce: “It is achieved either by means of a SIM card which retrieves device identity directly from the subscriber device, or by a direct link to core network elements.” There is no information on how the SIM sends the information, possibly a hidden SMS or USSD message is sent. That would mean that it is a combination of SIM distribution and the SMS or USSD solution. Obtaining information from core network elements must be equivalent to the network solution.

6 CMCC Self-registration

This chapter presents the design and implementation of the application based on the CMCC's requirements for self-registration, see appendix A. For description of the whole solution see section 4.3. The first two sections describe the Symbian operating system and the Symbian C++ language; they should be read in order to fully understand the implementation. However, this chapter should be considered as rather normative and readers should consider other sources in order to attain complete understanding of this operating system and programming language. The working environment (devices, tools, etc.) is also covered, apart from the language barrier (the English used at the office) which was an obstacle that had to be breached occasionally. The final design and implementation of the application is described in the last section of this chapter, complemented by the UML diagrams in appendix B.

6.1 Symbian OS

Symbian OS has its roots in Psion's EPOC software and is an operating system specially designed for mobile devices with all their limitations and possibilities. Associated libraries, user interface frameworks, and reference implementations of common tools are available in publicly documented APIs. The source code is not publicly available, but nearly all source code is provided to Symbian OS phone manufacturers such as Sony Ericsson. The source code was consequently available during the development phase of this thesis. The operating system runs exclusively on ARM processors due to their power saving features. Symbian OS is structured with pre-emptive multitasking, multithreading, and memory protection. Symbian programming is event-based and the CPU is turned off when applications are not directly handling an event, hence the battery life can be prolonged. Microkernel architecture is used, which means that only the necessary code, specifically the scheduler and memory management, is within the kernel. The networking and file system support are provided by user-side servers. In the system libraries there is a large networking and communication subsystem with three main servers: EPOC Telephony (ETEL), EPOC Sockets (ESOCK), and C32 (responsible for serial communication). Thus when an ETEL client establishes a connection with a communication device an appropriate module, called a TSY module, is loaded which contains the implementation for the corresponding device. The ETEL-abstracted API is then used to control the device. The user interfaces are maintained by other parties, such as UIQ, however the base classes and substructure for all UIs are present in the Symbian OS. [58]

The client-server model is used extensively to implement much of the Symbian OS functionality. The server runs in its own thread and executes commands received from the client (application). An example is the file system server which receives and executes commands to manage, create, delete, write, and read files. The model is not only used by the OS, but the Symbian OS also provides a client-server framework which programmers can use to implement their own client-server applications.

The central repository is a persistent data storage service from where common settings can be read and change notifications can be received. It comprises of a large number of repositories (2^{32}) which are identified by a unique ID (UID), and each setting is identified by a specific key, e.g. hence the software version of the phone software can be retrieved from this central repository. [57]

SIS files are standard way to distribute Symbian applications. After a SIS package has been created on a PC the application can be installed or used to update previous installations of the application by using connectivity software or the on-board installation program of the phone [48].

6.2 Symbian C++

Symbian C++ is a modified version of the C++ language specially designed for the Symbian OS. It has strong emphasis on conserving memory. Selected idioms and nonstandard characteristics of the Symbian C++ language are described in this section.

6.2.1 Symbian OS classes

There are four main categories of classes: data type, heap, resource, and interface classes, with the prefixes ‘T’, ‘C’, ‘R’, and ‘M’. Data type classes are values of a specific type encapsulated and hold methods for operations on these type of values, e.g. TChar, Tint, TBool, etc. Heap classes are instantiated on the heap and referenced by pointers. Heap classes are derived from *CBase* which ensures that the destructor is implemented and all data members are initialized to zero when instantiated. Resource classes are used when a resource is controlled by e.g. a server. The client or handle is then a resource class. For example when files are handled a resource class is used to open, manipulate, and close the file. Interface classes are abstract classes used to define an interface for other classes to use. [54]

6.2.2 Descriptors

Descriptors are a family of classes used for string and binary buffer handling. Descriptors are minimized for minimal overhead and prevent data from being written outside of the allocated buffer. Each descriptor holds length information, thus it does not need to be NULL-terminated and be used to store binary data as well as text, i.e. there is no need to scan for trailing null characters, or to allocate room for them. Descriptors can be constant, or non-modifiable, and modifiable with many modifying functions which allow string manipulation and calculation of maximum length. The type (identifies the underlying memory layout of the data it holds) is also defined in the descriptor. The “types” are various implementations of base abstract descriptor classes. The simplest are pointer descriptors which consist of length and address; and maximum length if it is modifiable, thus a pointer can be used to describe a buffer which is partially allocated. Buffer descriptors are stack based and contain the data as part of itself. The maximum length is set at compile time. Heap descriptors refer to data stored on the heap owned by the descriptor. The maximum length can be changed dynamically. Descriptors exist in either 8-bit ASCII or 16-bit Unicode format. Descriptors are widely used and are often passed as arguments in Symbian OS API functions. The use of descriptors is unique and can cause confusion for programmers new to Symbian OS. [51] [52]

6.2.3 Exception handling

Symbian handles Exceptions very differently compared to the try and catch mechanism in C++. Error handling and recovery are very important for limited resource devices. The phone should not crash or the user loose any data if an application runs out of memory. The mechanism provided by Symbian OS is extensive and used heavily in Symbian OS software; it comprises a significant portion of the design and development effort. The mechanism is not only used to handle errors and perform recovery, it is also commonly used as a call back mechanism. Symbian OS uses status codes which are returned when a function is called. The codes can indicate success or hold some kind of error code. However, not all return codes are tested by the programmer and the calling function might not know how to handle particular errors. To solve this problem an exception-based error handling and recovery mechanism based on leaves and traps is provided. When an error occurs a leave is invoked and the function exits immediately. The control then returns to the calling function, if no trap exists, it also exits at that point. This continues and stops when the first trap which handles the error is encountered. Unlike simple return codes, a leave cannot be ignored. The trap is a macro which takes two arguments, the function which is to be invoked and a variable where the return code is stored. When the function call returns, the return code can be examined and proper measures taken. A function acts as if a return occurred when it exits due to a leave, thus all automatic variables will deallocated. However, if a cleanup that requires explicit code such as delete statements will be skipped if it occurs after the leave, then the cleanup stack is used to support this. Automatic pointer variables are pushed into this stack before a section where a leave can occur, and explicitly popped when there no longer is a risk. In case there is a leave, the pointers will be popped automatically. There is a Symbian OS convention that all functions that might leave shall have an ‘L’ suffix, and a ‘LC’ suffix if a pointer also is pushed into the cleanup stack (it still has to be popped). More complex cleanup is sometimes necessary and several cleanup functions are available. The reason why the throw and catch mechanism from C++ were not used is because they was not a part of C++ at the time Symbian OS was written. Additionally, the Symbian OS mechanism is more lightweight and efficient. [53]

The constructor in C++ is called immediately after the *new* operator function, thus the programmer has no chance to push the pointer to the allocated memory to the cleanup stack. The two-phase constructor solves the problem with a rather simple concept: an extra method is supplied which completes the object construction. It is called from the constructor and a leave can occur since it is a normal function. [53]

6.2.4 Active objects

An Active Object is a Symbian OS idiom by which one process or thread can request services from another. Applications typically use this idiom when accessing system services such as telephony. Nearly all Symbian OS applications are based on this. It is a client-server relationship and a thread or process that provides the services is called an asynchronous service provider. A function call from a client application returns immediately, but it only means that the request has been dispatched, the requester is informed when the request is completed through a signaling mechanism. An asynchronous request status is passed as a parameter and the service provider stores a success or error code in the request status when the request is completed. The provider signals to the requester that a request is completed through a request semaphore. The active object encapsulates the general behavior of making requests to an asynchronous service provider and handles the completion of requests. An application commonly is comprised of several active objects; the active scheduler controls the handling of asynchronous requests from active objects and there is one active scheduler per thread. Active objects and the active scheduler provide a system of non pre-emptive multi-processing which runs on a single thread. Multi-processing in conventional system is often done using multiple threads. Compared to a system with co-operating threads, a system with active objects is significantly more effective since the run-time cost of an active object is less than that of a thread, furthermore the creation and destruction of active objects is considerably more effective than creation and destruction of threads. Additionally, in Symbian OS, it is a lot easier to write systems based on active objects than system based on threads. [49] [50]

6.3 Smartphones

The mobile phones used during development were Sony Ericsson smartphones, model M600 and W950 with Symbian OS 9.1 and UIQ 3 as graphical user interface. Both have touchscreens with an accompanying stylus; can operate in GSM, GPRS, and UMTS networks; utilize Bluetooth, USB, and IR data transmission; support MMS, SMS, USSD, etc. The W950 has a built in flashmemory of 4GB and the M600 a memory stick slot.

Flight mode is a functionality available in these phones that allows the user to turn of the transmitting of radio signals from the phone, i.e. transform the phone to a regular palm-top. The option can be chosen on startup or from a menu item. The purpose of this is to make it possible to use the functionalities which don't transmit radio signals in sensitive locations such as aircraft or hospitals.

6.4 Tools

The Codewarrior development environment [18], dedicated for embedded applications, was used. It consists of a source code editor, compiler, and debugger. Customized emulators for each model phone under development were available and floating licenses were used which caused interruptions in the development work. Source Insight [47] was used as a support tool; it was primarily used as a code viewer, but is also a code editor. Source Insight parses the source code and maintains a database of symbolic information dynamically, and presents contextual information automatically. Reference trees, class inheritance diagrams, and call trees are presented in a clear manner.

A tool called Bypass started to be used relatively early in the development phase. It consisted of a program with which a phone could be flashed, a USB-cable, and a configured development environment. By using Bypass the communication abilities of the phone could be used via the USB-cable on the emulator, i.e. calls could be made and SMS messages sent through the emulator's GUI. The usage of Bypass shortens the time considerably when the code has to be tested on target (i.e. on an phone). Work flow without Bypass:

Edit code – Compile – Build – Create SIS – Put the SIS in the phone – Install SIS – Run

The process was of course eased by use of batch-files; however putting the SIS in the phone was somewhat cumbersome since the USB-cable was used to collect logs. A reconfiguration had to be made in order to transmit the SIS via USB; hence Bluetooth or a memory stick was often used. By using Bypass the work flow was shortened to:

Edit code – Compile – Build – Run

There was a test-server available provided by Bitfone. The existence of the test-server was not known until rather late in the development phase, however, this proved to be more of an advantage than a disadvantage. The server used for "real registrations" was operational, although it only answered with registration failure messages since the models under test were not in the database. The test-server did not provide any better service in that sense since it only sent registration success messages, if there was a registration success. There was a possibility to receive the log from the server in order to investigate the cause of a registration failure; however, an appointment had to be made well in advance with a rather busy person working at Bitfone who would monitor the log during a limited period of time. Information was exchanged via e-mail and it took between 10 and 20 minutes from when a registration message was sent to an e-mail containing the log being received. It proved almost impossible to receive the correct log without an appointment. The test-server could be down sometimes and the address changed without advanced notice. The test-server required 8-bit encoding of the message although it was explicitly stated in the requirements that hexadecimal encoding must be used, see requirement 13 in appendix A. The real server answered only to messages using hexadecimal encoding, thus a macro had to be used so that function testers could switch to 8-bit encoding and test the functionality with the incorrect (but acceptable) format. The content of the message also caused some confusion, probably because of the inconsistency of the requirement, see requirement 13 in appendix A. It was assumed that an example of the content was:

"351254008472118 21/Sony Ericsson/V780/Ver2.1.0.18"

The test-server required:

"IMEI: 351254008472118 21/Sony Ericsson/V780/Ver2.1.0.18"

The real server accepted both formats of the content.

An application which simulated the server (described in detail in section 6.5) was developed at an early stage of the development phase and proved considerably more effective than the Bitfone test-server. Additionally, a substantial number of the test cases can not be tested by using only the test-server, see section 6.6.1.

Baselines (source code and binaries with specific versions of the phone software) had to be downloaded occasionally to test interoperability. Command prompt and GUI tools were available to download a baseline and add, update, or remove software components.

Databases containing problem reports, change requests, and various documents were utilized in the search for information, as well as search-tools while searching for relevant code.

6.5 Additional applications

Two additional applications had to be utilized in conjunction with the development of the main application. Both were already available, but they had to be adjusted and extended in order to serve the purposesnecessary in this development effort. One facilitated development and the other one tested the main application.

The facilitating application was originally developed by programmers at the local office. Its original purpose was to test GUI-applications and consisted of a menu by which a specific application (process) could be started. The process could then be killed, the code changed and recompiled (if the emulator or Bypass were used), and run again without the need to create a SIS file. However, the processes were killed by using their GUI, and the self-registration application had no GUI. An additional option had to be added which found and killed the process, and that was the only adjustment needed. The application had the rights to kill the self-registration application process since it was its child process. The same thing can be done by using an available shell-console, but it is more convenient to use a GUI-application, especially when testing on the target.

The test application was also originally developed in-house and its purpose was to test the functionality of software modules, e.g. test if an mp3-file was played correctly below the GUI. The application could be installed on the phone and provided a console by which numbered commands could be chosen. The application was used at the early stage of the development phase, before the existence of Bypass was known, to test functions on target, e.g. retrieve the IMEI. Later it was used to simulate the Bitfone-server; programmed to listen for registration messages and send acknowledgement messages. It proved to be more efficient then testing on the Bitfone test-server, and was a necessity since the Bitfone test-server did not cover all test cases.

6.6 Implementation

My knowledge of C++ and Symbian was somewhat limited when the implementation work began; however, the aim was to implement the application as soon as possible. This resulted in a short design phase and a form of experimental programming, opposite to the typical waterfall model. The work flow followed can be compared to extreme programming where several designs were implemented and later set aside when new designs were followed up. One of these (the synchronous implementation) is described in this section. The final design ended in a Symbian standard design which proved superior to the others, although it required a considerable period of study. The implementation was more time consuming than expected, the conventional wisdom is that it takes three times longer to develop applications with Symbian C++ than regular C++. The implementation work has been a valuable experience and has enhanced my understanding of the industry.

6.6.1 Test cases

Test cases had to be defined in order to test the application during development and support the function testers. Only the purpose of each test case is defined below, i.e. what is tested, not how. Test cases that could not be tested by only using the test-server are marked with a ‘*’.

Table 2. Test cases

Test case #	Test case
1	Application starts at startup.
2	Application is terminated if not CMCC SIM
3	SMS message is not sent if there is an IMSI match.
4	Application wait until the CMCC network is available
5	SMS message is sent with correct information
6*	Application listens for messages through the whole execution
7	The acknowledgement message is caught
8	IMSI stored if success message
9*	IMSI not stored if failure message
10*	IMSI not stored if wrong sender address in ack. message
12*	IMSI not stored if wrong IMSI in ack. message

In order fully fulfill the requirements, see appendix A, additional cases must be considered that were not directly based on the requirements. They might seem remote, however combined altogether they covered cases that impose a threat to the quality of the application.

The registration message must be sent as soon as possible (it is a vital part of the Device Management System), which mean that hindering of the sending by user interaction; network instability, or phone limitations must be avoided. Basically it means that the message is re-sent after a sending-failure and the application waits until the correct network is available. User interaction can occur, for example flight mode is entered during the sending process, and thus there must be a way to recover from this.

There is a risk that an acknowledge message is received before the registration message is sent, e.g. if the user turns off the phone after the registration message is sent, but before the acknowledgement message is received. Several acknowledgment messages might also be sent if the user turns off the phone several times before an acknowledgement message is received or the SMSC resends a message due to signaling errors. Since the user must not be exposed to these messages, these cases must be considered and the needs to application start listening for messages **before** the registration message is send, and continue listening after the first acknowledgement message is received, i.e. all the time.

6.6.2 Synchronous implementation

After the first analysis of the requirements, the application seemed rather simple and the goal was to make it very simple. The decision was made to make it synchronous with minimal use of active objects. The flowchart below describes the intended implementation.

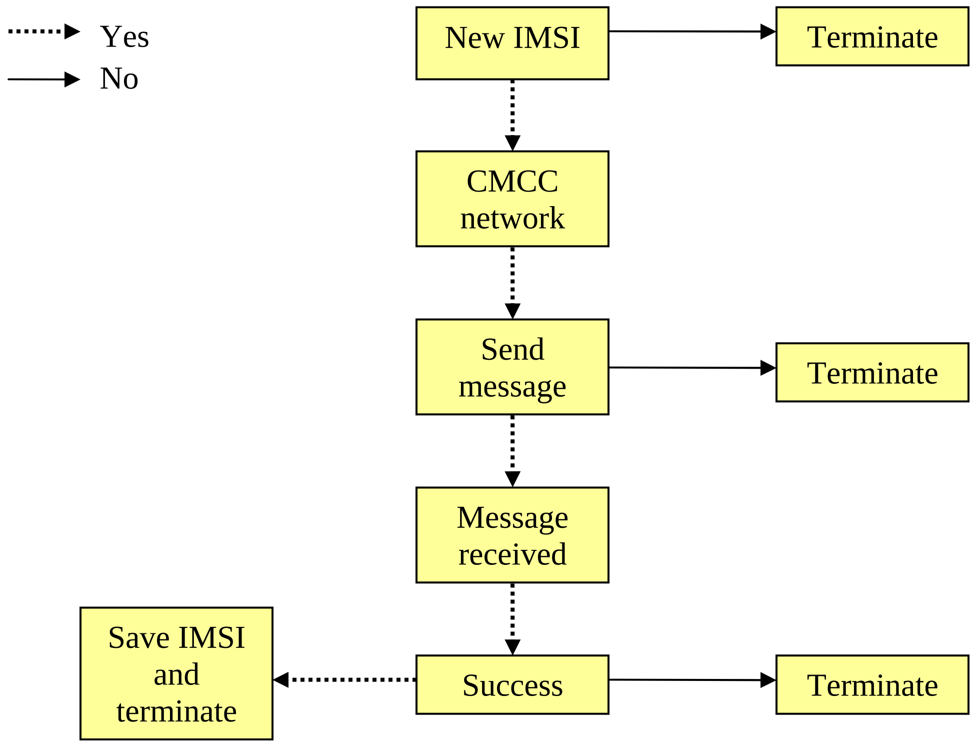


Figure 2. Synchronous implementation flow chart.

The application would start by comparing the IMSI on the SIM with the one stored on the phone, if there is a mismatch the application would continue by waiting on the CMCC network, otherwise it would terminate. A rather complicated algorithm had to be implemented to make sure that the application did not continue until the CMCC network was available. The same algorithm was also implemented in the final application and is described in the section 6.6.3.1. The SMS message could then be assembled and sent. If there is a sending failure, then the application will simply terminate and the application will try to send a new message on next power up. The next step is to start listening for acknowledgement messages. Upon arrival the message would be parsed, and success or failure determined. Success would mean that the IMSI was saved and failure termination of the application.

This seems to be a simple and fairly good solution, the use of active objects is minimized and the execution is asynchronous. The application would fulfill the requirements to large extent, but was not perfect. The SMS is not re-sent in the even of a sending failure and the solution does not solve the problem of what to do when several acknowledgement messages are sent (as described in section 6.6.1). The network should be checked before a re-sending attempt is made (in case the phone user has chosen to enter flight mode or the network connection is lost) and the application must listen for messages throughout its entire execution. A considerable amount of time was spent in an attempt to keep the application simple, but still cover all cases. However, it was set aside and a standardized solution based on active objects was eventually pursued. It proved far easier to work with and did not violate the Symbian coding standard.

6.6.3 Final design

The final design consists of an engine which is initiated and started after it is determined that a CMCC SIM is inserted in the phone. The engine is based on the active object (and active scheduler) framework and makes use of the observer pattern (callbacks) [55]. The execution is controlled in the engine by switching states in the *RunL* of the engine (the *RunL* can be described as a while-loop that waits for function calls to return and can be activated or de-activated). One of the states is “abortion of registration”, which results in deletion of all unnecessary objects. The only remaining objects after abortion are those needed for handling of incoming messages. The motive is to free as much memory as possible, as soon as possible. The abortion occurs when a registration success message is received, there is an IMSI-match, or necessary information for the registration message could not be obtained. In order to gain access to telephony services, a polymorphic interface dynamically loaded library (the TSY module) is loaded. A socket is used when listening for messages and starting listening is among the first operations executed (already in the second phase constructor of the engine). After an IMSI-mismatch the RunL is activated and the application starts waiting for the CMCC network (see section 6.6.3.1 for detailed design). When the correct network becomes available the execution proceeds with sending of the message. The sending function is static; this is due to simplicity reasons and to make sure that everything is re-done in case the message needs to be re-sent. It is somewhat against Symbian coding standards, but the application does not need to handle events in real time (from a GUI) and an incoming message can still be caught. The application then proceeds into a state where it only listens for incoming messages and the *RunL* is de-activated. When a message arrives, the SMS watcher (baseline component) passes it to the socket listening on the specific SMS-port. In this way the application catches the message and proceeds by parsing it. The engine is informed via an observer notification if an activation success message has been received. The IMSI is stored in a txt-file in the private directory of the application and checked again on every startup. In case there is a sending failure of the message, the engine is informed via an observer notification and start a timer. After the timeout (ten minutes) the *RunL* is re-activated and starts by waiting for the CMCC network (since lack of connectivity to the network can be the root cause of the sending failure), then re-create the message and resends it.

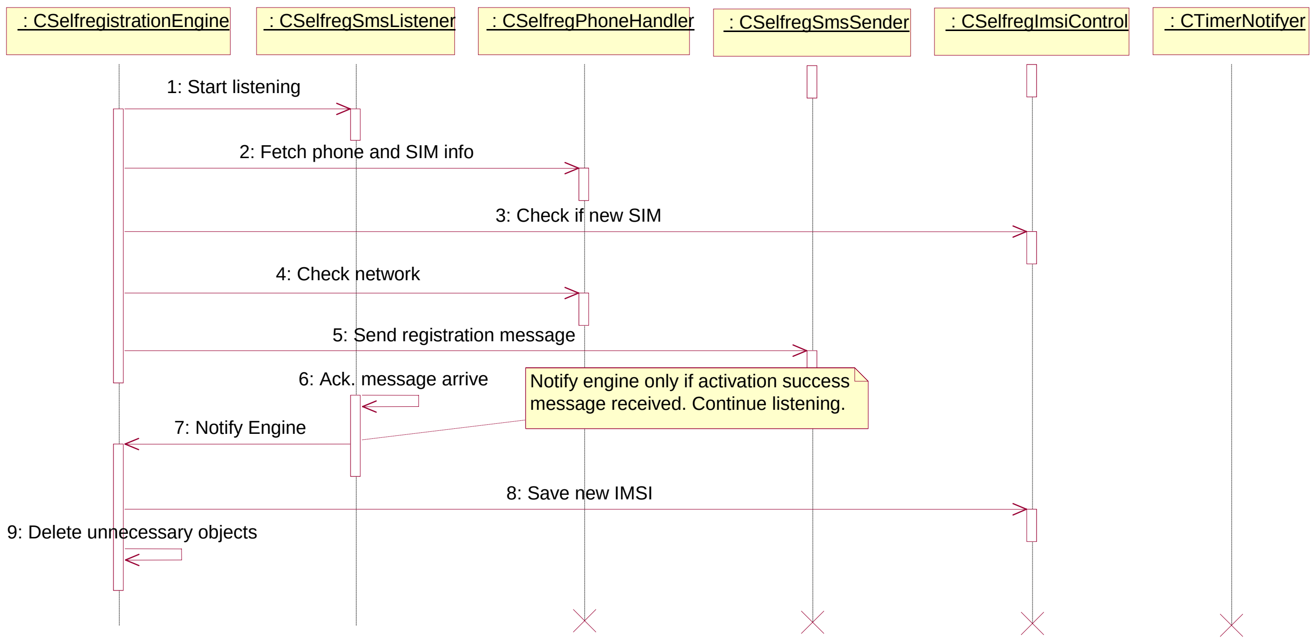


Figure 3. Main execution

Before the engine is constructed the application first checks if a CMCC SIM is used, the application exits if it is not a CMCC SIM. No exceptions are described in the diagram shown in figure 3.

6.6.3.1 Waiting for the network

The algorithm used for waiting on the CMCC network was somewhat cumbersome to program since the API did not provide any notification service for changes of specific network status entities, such as the current service provider, only if there were a change in the network status, thus the application can be notified if something has changed, but not what. Therefore, the MNC and MCC had to be checked each time a change notification came in. Since the network status could change during the checking, a notification request had to be made directly after a network status change notification. A cancellation of the notification request had to be made in case the CMCC network became available. It was a bit lengthy to make this work according to good Symbian coding standards; however it resulted in a thorough understanding of the Symbian C++ language. The algorithm also deals with the flight mode case.

6.6.4 Dependencies

Figure 4 illustrates the following relationships:

- Application is started on every start-up, this is made possible by adding the component to the startup configuration.
- Application uses the phone module (TSY) to gather and check phone information.
- Application gets the current software version from the central repository.
- Application connects with the socket server in order to send and receive SMS messages via sockets.
- The file server enables the handling of the txt-file used for storing and retrieving the IMSI.

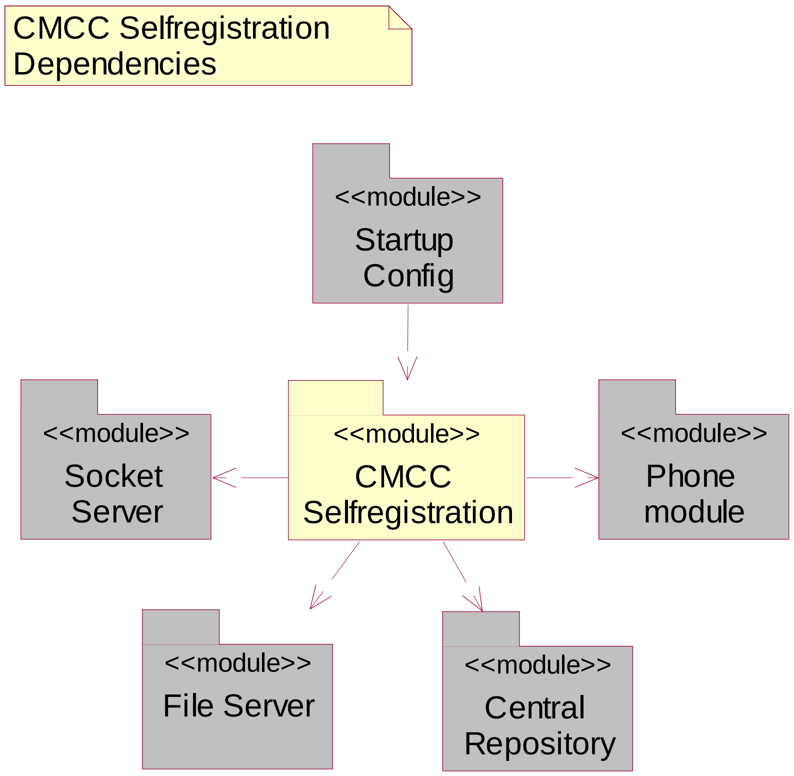


Figure 4. Dependencies

6.6.5 Error Handling

The standard Symbian OS leave mechanism, in conjunction with the cleanup stack and trap harness is used to handle runtime errors.

Errors are expected to be raised under the following fault conditions:

- Out of memory
- Out of disk space
- Read/Write file failure
- Socket server failure
- File server failure
- Sending of SMS message failure

6.6.6 Fault and Recovery

Fault conditions are presented in headlines followed by recovery description. Termination of the application will occur if the constructions of necessary components on startup fail.

The registration can under some conditions be aborted, which results in the SMS message not being sent and deletion of unnecessary components. The application will keep listening for acknowledgement messages in order to prevent the user from being exposed to them.

Abortion of registration will be triggered under these conditions:

- The stored IMSI is identical to the one on the SIM currently in use
- Activation success message is received
- Failure to gather the necessary phone information for the SMS message to be sent

6.6.6.1 Out of memory

If not enough memory is available on startup the OS will attempt to re-acquire memory from background applications. If this fails the application will not start and the user will not be notified. During runtime these errors will be handled either by abortion of registration or re-sending of SMS message. If it happens during the construction of the engine the application will exit.

6.6.6.2 Out of disk space

The errors can only occur when handling the txt-file, therefore they are equivalent to the Read/Write file failure condition.

6.6.6.3 Read/Write file failure

If an error occurs when reading from the file it will be handled by assuming that is the first startup, i.e. by starting the registration. If an error occurs when writing to the file nothing will happen until the next startup when the error will be handled implicitly because a reading error occurs or there is a mismatch in the IMSI comparison. The old file is deleted when a new IMSI is stored.

6.6.6.4 Socket server failure

If an error occurs when the application starts listening for messages the application will exit. If it happens when the message is to be sent, the engine will be notified and it will try to re-connect and re-send the message. If the connection listening socket connection is lost it should try to re-connect as soon as possible.

6.6.7 Open issues

The application is finished and operational; however some open issues still remain. Decisions have been taken regarding all open issues described in this session, although uncertainty remains.

6.6.7.1 Continue listening for messages

The program could end completely after receiving an activation success message. Presently, the application keeps listening for messages in order to avoid exposing them to the user. Several messages can under rare circumstances be sent out to the device, e.g. if the user restarts the phone several times after the registration message is sent, but before the acknowledgement message is received. This combined with malfunctioning network or phone could cause the problem mentioned above. The application checks if an acknowledgement message is received before a registration message is sent, and the registration is aborted if a success message is received. Considering this, the chance that the user will be exposed to an acknowledgement message is very low.

6.6.7.2 Re-sending of SMS message

There could be a limit to number of tries to re-send the SMS message. Presently, in most cases, the application will try to re-send the message until it succeeds. The application is designed this way to avoid hindering of registration by user interaction.

6.6.7.3 Occupying the SMS port

The specified SMS port (see requirement 11 in appendix A) is occupied by this application and no other application can use it. According to ETSI GSM 03.40 [56], the port should be available to third party applications.

Table 3. Ports

VALUE	ALLOCATED
0-15999	As allocated by IANA (http://www.IANA.com/)
16000-16999	Available for allocation by applications
17000-65535	Reserved

The risk that this will result in a problem is very low, and no other suitable solution has been found yet, however it should be possible to catch the message on some lower level. The problem can be mitigated if the application stops listening for messages after it has received an activation success message. Presently, the application keeps listening for messages in order to avoid exposing them to the user.

6.7 Evaluation of solution

The application implemented is robust and it is impossible to avoid registration, furthermore the user can not be exposed to acknowledgement messages. It has been made possible on the cost of decreased phone performance (a few kB of memory) since the application runs all the time. Malfunctioning phone or network may hinder registration, thus it can be assumed that very close to all mobile phones will be registered within their lifetime. Support centers may easily help users to register (receive updates) by simply telling them to restart the phone. Most phones will accomplish registration within minutes of first being used. How many messages will be sent by each handset differ much, however, it can be assumed that at least three registrations will be executed during the lifetime of a mobile phone. Users may sell their phone on second hand or borrow the phone temporarily to a friend, which is the equivalent of at least six SMS messages sent between the registration server and the mobile phone.

7 Conclusions

The most suitable solution, or at least the easiest, is the SMS solution, although Swapcom has shown that the network solution can and is used. The self-registration has unexpected positive side-effects in the form of guarantee time references and improved tracking of stolen phones. A good solution to a problem such as the Device Management registration can not only rely on purely technical features, it is a rather complex system with many actors that need to cooperate in order to provide the best user experience. Business aspects must be addressed, and there is a comfort in using well-known technologies instead of trying new innovative solutions. The thesis indicates that it can be preferable to retrieve the relevant information from devices instead of retrieving the same information already available in the nodes of the network. Partly, this is due to limitations in the standardization specifications, e.g. the IMEISV holds only two digits reserved for the phone software version.

The importance of proper preparations and long design phase has been valuable lessons learned during the development phase. Additionally, not constrain the work to directly available tools and resources but look for other solutions is an important advice to remember.

8 Future work

It is rather hard to say which of the solutions will prevail, or what new technology will facilitate the registration. However it is certain some kind of solution will be needed for the registration problem. Solutions are overall not presented to the public, since it often is a business secret, hence it will be rather hard to follow the development if not explicitly involved in it. Probably, we will not notice anything until a software update request pops up on our mobile phone, without knowing for certain when and how the registration was executed.

There is a possibility that SIM-cards will not be used in the future, instead users will personalize their phones by personal codes. It would dramatically change the whole system, but the self-registration could easily be adjusted, the registration would be executed when the user performs the personalization.

References

[1] Redbend software, *Key Criteria for Evaluating Technologies for Effective Firmware Over-The-Air (FOTA) Updating of Mobile Phones.* Page1. http://www.redbend.com/pdf/RedBend_FOTA_WP.pdf
Last access on 2007-03-21

[2] CDG Technology Forum on Application Security, *Advantages Of Firmware Over The Air (FOTA) For Mobile Security*.
http://www.cdg.org/news/events/CDMASeminar/050513_Tech_Forum/2-OTAFF_051305.pdf
Last access on 2007-03-21

[3] Sony Ericsson, *Update Service*.
http://www.sonyericsson.com/spg.jsp?cc=gb&lc=en&ver=4000&template=ps7&zone=ps&fid=21839&numItems=18
Last access on 2007-03-21

[4] Cybercom Group, *Always updated with Sony Ericsson*.
http://www.cybercom.se/templates/Page____793.aspx?epslanguage=EN
Last access on 2007-03-21

[5] Wikipedia.
http://en.wikipedia.org/wiki/2G
Last access on 2007-03-21

[6] Wikipedia.
http://en.wikipedia.org/wiki/3G
Last access on 2007-03-21

[7] Wikipedia.
http://en.wikipedia.org/wiki/General_Packet_Radio_Service
Last access on 2007-03-21

[8] GSM World, *China Mobile - Network Information.*
http://www.gsmworld.com/roaming/gsminfo/net_cnct.shtml
Last access on 2007-03-21

[9] Wikipedia.
http://en.wikipedia.org/wiki/SMS
Last access on 2007-03-21

[10] Wikipedia.
http://en.wikipedia.org/wiki/USSD
Last access on 2007-03-21

[11] TDSCDMA Forum, *Service Research and Feature Applications of TD-SCDMA.* Page 1.
http://www.tdscdma-forum.org/EN/pdfword/200651715435494490.doc
Last access on 2007-03-21

[12] Bladox, *SIM Toolkit*. Network commands.
http://www.bladox.com/devel-docs/gen_stk.html
Last access on 2007-03-21

[13] Redbend software, *Phenomenal growth of FOTA phones*. Center of page.
http://www.redbend.com/
Last access on 2007-03-21

[14] Traud, *What is OMA SyncML DS* .
http://www.traud.de/gsm/SyncML.htm
Last access on 2007-03-21

[15] OMA, *Enabler Release Definition for Firmware Update ManagementObject*. Page 6.
http://www.openmobilealliance.org/release_program/fumo_v1_0.html
Last access on 2007-03-21

[16] OMA, *Firmware Update Management Object Architecture*
http://www.openmobilealliance.org/release_program/fumo_v1_0.html
Last access on 2007-03-21

[17] OMA, *Device Management Requirements*
http://www.openmobilealliance.org/release_program/fumo_v1_0.html
Last access on 2007-03-21

[18] Freescale, *Codewarrior*
http://www.freescale.com/webapp/sps/site/homepage.jsp?nodeId=012726
Last access on 2007-03-21

[19] OMA, *About Open Mobile Alliance*.
http://www.openmobilealliance.org/about_OMA/index.html
Last access on 2007-03-21

[20] EventHelix, *GSM GPRS Sequence Diagrams.*
http://www.eventhelix.com/RealtimeMantra/Telecom/#GSM_GPRS_Sequence_Diagrams_
Last access on 2007-03-21

[21] Wikipedia.
http://en.wikipedia.org/wiki/GPRS_Core_Network
Last access on 2007-03-21

[22] 3GPP TS 22.016, *International Mobile Equipment Identities (IMEI).* 2006-12-14

[23] 3GPP TS 23.003, *Numbering, addressing and identification.* 2006-12-19

[24] BABT, *IMEI allocation*. Background section.
http://www.babt.com/gsm-imei-number-allocation.asp
Last access on 2007-03-21

[25] Slashphone.
http://www.slashphone.com/64/5740.html
Last access on 2007-03-21

[26] Wikipedia.
http://en.wikipedia.org/wiki/Imei
Last access on 2007-03-21

[27] Wkipedia.
http://en.wikipedia.org/wiki/Type_Allocation_Code
Last access on 2007-03-21

[28] 3GPP TS 22.090: *Unstructured Supplementary Service Data (USSD).* Section 6.
Last access on 2007-03-21

[29] G. Q. Maguire Jr., *Mobile and Wireless Network Architectures*. Page 196-197.
http://www.it.kth.se/courses/2G1330/Lectures-2007/MWA-20070117-tagged.pdf
Last access on 2007-03-21

[30] Wikipedia.
http://en.wikipedia.org/wiki/Mobile_Application_Part
Last access on 2007-03-21

[31] Cellicium, *USSD in the GSM Standard.*
http://www.cellicium.com/ussd/info/
Last access on 2007-03-21

[32] ViaMedia, *USSD Billing*.
http://www.viamedia.co.za/news.htm
Last access on 2007-03-21

[33] Forum Nokia, *Series 40 Platform 3rd Edition: OMA Device Management.*
http://sw.nokia.com/id/51a256f9-7174-4dc9-84b5-07302269025d/Series_40_Platform_3rd_Ed_OMA_Device_Management_v1_0_en.pdf
Last access on 2007-03-21

[34] S. Bradner, *"Key words for use in RFCs to Indicate Requirement Levels"*,
Request for Comments: 2119, IETF, Network Working Group, March 1997
http://www.ietf.org/rfc/rfc2119.txt
Last access on 2007-03-21

[35] Wikipedia.
http://en.wikipedia.org/wiki/Original_Equipment_Manufacturer
Last access on 2007-03-21

[36] Bitfone Corporation, *Customers.*
http://www.bitfone.com/usa/partners_customers.shtml
Last access on 2007-03-21

[37] Insignia, *Customers.*
http://www.insignia.com/content/about/customers.shtml
Last access on 2007-03-21

[38] Synchronica, *Customers.*
http://www.synchronica.com/syncml-partners/syncml-customers.html
Last access on 2007-03-21

[39] Wikipedia.
http://en.wikipedia.org/wiki/Communications_Assistance_for_Law_Enforcement_Act
Last access on 2007-03-21

[40] Wikipedia.
http://en.wikipedia.org/wiki/PLMN
Last access on 2007-03-21

[41] Isabelle Chan, ZDNet Asia, *Report: SMS traffic to double in AP by 2010.*
http://www.zdnetasia.com/news/communications/0,39044192,39252956,00.htm
Last access on 2007-03-21

[42] Swapcom, Catalogue, *Mobile Software Architect.* Page 8 and 10.
http://www.swapcom.fr/pdf/solutions-2006/catalogue/catalogue-swapcom-2006.pdf
Last access on 2007-03-21

[43] Swapcom, *Device Management Center*. Automatic Device Discovery.
http://www.swapcom.fr/mobile-solutions/device-management-center.htm
Last access on 2007-03-21

[44] OMA, *Device Management Working Group.*
http://www.openmobilealliance.org/tech/wg_committees/dm.html
Last access on 2007-03-21

[45] Bilal Siddiqui, DevX, *Compressing XML—Part I, Writing WBXML*.
http://www.devx.com/xml/Article/16754/0/page/1
Last access on 2007-03-21

[46] Symbian, *Symbian Developer Library*. Using CDMA SMS Messaging.
http://www.symbian.com/developer/techlib/v9.1docs/doc_source/guide/Messaging-subsystem-guide/index.html
Last access on 2007-03-21

[47] Source Insight, *Source Insight Program Editor and Analyzer*.
http://www.sourceinsight.com
Last access on 2007-03-21

[48] Eric Bustarret, NewLC, *How to create a SIS file.*
http://www.newlc.com/How-to-create-a-SIS-file.html
Last access on 2007-03-21

[49] Symbian, *Symbian Developer Library*. Asynchronous Services Overview http://www.symbian.com/developer/techlib/v70sdocs/doc_source/devguides/cpp/Base/InterProcessCommunication/AsynchronousServicesOverview.guide.html
Last access on 2007-03-21

[50] Symbian, *Symbian Developer Library*. Introduction to Active Objects Scheduler. http://www.symbian.com/developer/techlib/v70sdocs/doc_source/DevGuides/cpp/Base/InterProcessCommunication/AsynchronousServicesGuide/AsynchronousServicesGuide3/IntroductionToActiveObjectsScheduler.guide.html
Last access on 2007-03-21

[51] http://www.symbian.com/developer/techlib/v70sdocs/doc_source/DevGuides/cpp/Base/BuffersAndStrings/DescriptorsOverview.guide.html#BuffersAndStringsOverview%2eDescriptorsOverview%2emain
Last access on 2007-03-21

[52] Steve Babin. *Developing Software for Symbian OS. An Introduction to Creating Smartphone Applications in C++*. Dec. 2005. Section 6, Descriptors.

[53] Steve Babin. *Developing Software for Symbian OS. An Introduction to Creating Smartphone Applications in C++*. Dec. 2005. Section 4.5, Exception Error Handling and Cleanup.

[54] Steve Babin. *Developing Software for Symbian OS. An Introduction to Creating Smartphone Applications in C++*. Dec. 2005. Section 4.4, Symbian OS classes.

[55] Somenath Mukhopadhyay, NewLC, *Observer Pattern in Symbian application.* http://www.newlc.com/Observer-Pattern-in-Symbian.html
Last access on 2007-03-21

[56] GSM 03.40 version 5.8.1, *Technical realization of the Short Message Service (SMS)*, section 9.2.3.24.4

[57] Symbian, *Symbian Developer Library*. CRepository. http://www.symbian.com/developer/techlib/v9.1docs/doc_source/reference/reference-cpp/N102E6/CRepositoryClass.html#%3a%3aCRepository
Last access on 2007-03-21

[58] Wikipedia. http://en.wikipedia.org/wiki/Symbian
Last access on 2007-03-21

[59] Wikipedia. http://en.wikipedia.org/wiki/Base_Station_Subsystem
Last access on 2007-03-21

Abbreviations

API	Application Programming Interface
ARM	Advanced RISC Machine
ASCII	American Standard Code for Information Interchange
BSS	Base Station Subsystem
CMCC	China Mobile Communications Corporation
CPU	Central Processing Unit
CSD	Circuit Switched Data
DM	Device Management
FOTA	Firmware Over The Air
FUMO	Firmware Update Management Object
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
HLR	Home Location Register
ICCID	International Circuit Card ID
IMEI	International Mobile Equipment Identity
IMSI	International Mobile Subscriber Identity
IR	Infra Red
ITU	International Telecommunication Union
ITU-T – ITU	Telecommunication Standardization Sector
MAP	Mobile Application Part
MCC	Mobile Country Code
MMI	Man Machine Interface
MNC	Mobile Network Code
MSC	Mobile Switching Centre
MSISDN	Mobile Station International Subscriber Directory Number
NSS	Network and Switching Subsystem
ODM	Original Design Manufacturer
OEM	Original Equipment Manufacturer
OMA	Open Mobile Alliance

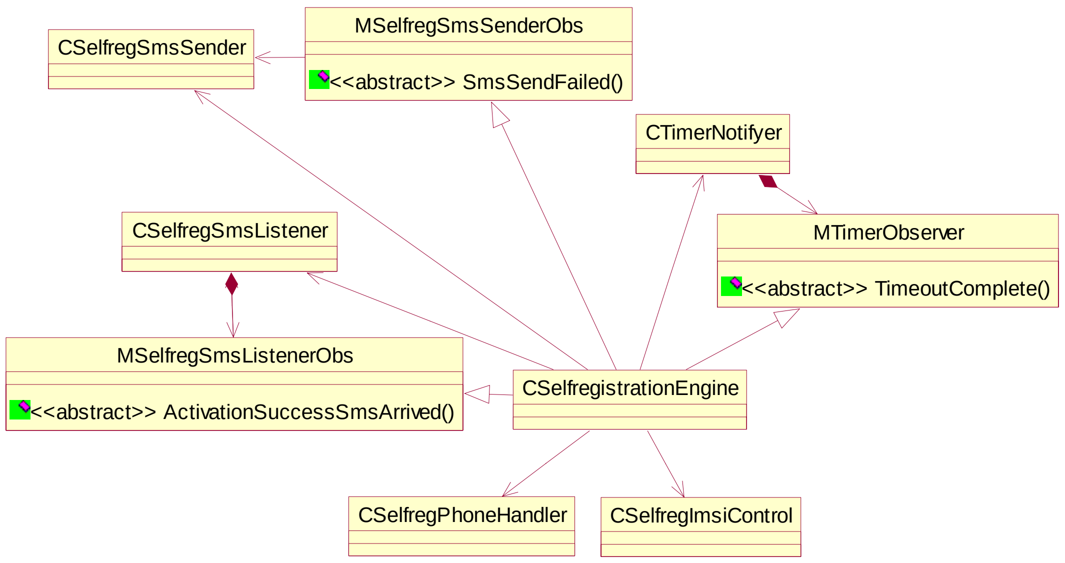
OMA	Open Mobile Alliance
OS	Operating System
OSE	Operating System Embedded
R2R	Ready to Run
SGSN	Serving GPRS Support Node
SIM	Subscriber Identity Module
SMS	Short Message Service
SMSC	Short Message Service Centre
SNR	Serial Number
SS7	Signaling System #7
TAC	Type Allocation Code
TMSI	Temporary Mobile Subscriber Identity
UIQ	User Interface Quartz
USB	Universal Serial Bus
VLR	Visitor Location Register
WAP	Wireless Application Protocol
WBXML	WAP Binary XML
XML	Extensible Markup Language

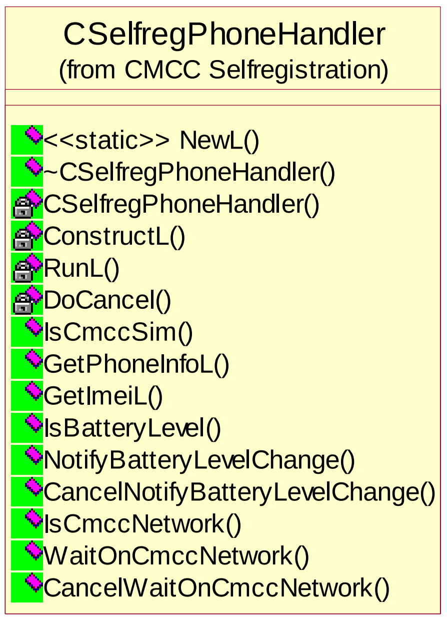
Appendix A – CMCC Self-registration requirements

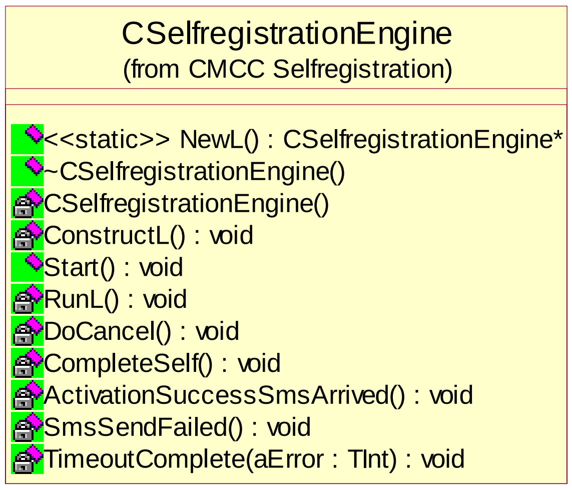
	Terminal self-registration
1	Terminal self-registration means that a terminal compliant with DM platform specification shall, at the first power-on, send terminal IMEI and other basic information to DM platform in SMS. The platform is responsible for parsing the SMS and establishes a list of terminal IMEI and MSISDN mappings. In case the user changes SIM, the terminal shall resend self-registration SMS to DM platform again.
2	At the first power-on, the terminal shall send a self-registration SMS to DM platform. SMS shall contain terminal IMEI, manufacturer name, device model, and software version. After sending self-registration SMS, terminal will be at normal standby status.
3	If terminal receives self-registration acknowledgement SMS from DM platform, it shall parse the messages correctly to determine success or failure of the activation. In the case of success, the terminal shall permanently store the IMSI information of the SIM card at a predefined position of the terminal (value of the position shall be mark bit used exclusively by terminal self-registration function. It must not be modified by other parts of the terminal).
4	Terminal shall, each time when restarts, check whether IMSI of SIM card is consistent with the IMSI stored on the terminal. In the case of inconsistency, the terminal shall resend self-registration SMS to DM platform.
5	Only SMS from DM self-registration number is trusted DM acknowledgement SMS.
6	DM Self-registration SMS (upward) and Self-registration acknowledgement SMS (downward) shall use CMCC defined port.
7	Terminal shall judge whether the SIM card is China Mobile's SIM card. Terminal must not send self-registration SMS for non-CMCC SIM.
8	The terminal self-registration process is hidden to the user. It must not display any information to user or provide the user with any operation interaction.
	Self-registration exception handling
9	At the first registration, the registration message is sent correctly, but no acknowledgment message is received from DM platform. If the first self-registration fails, IMSI of the SIM card shall not be saved on the terminal. Terminal shall initiate self- registration at next power-on.
10	When detecting a change of SIM card by the user, the terminal shall send a new self-registration SMS to DM platform. In case no acknowledgment message is received from DM platform, the terminal shall abort self-registration. If self-registration fails, IMSI of the SIM card shall not be stored on the terminal. Terminal shall initiate self-registration at next power-on.

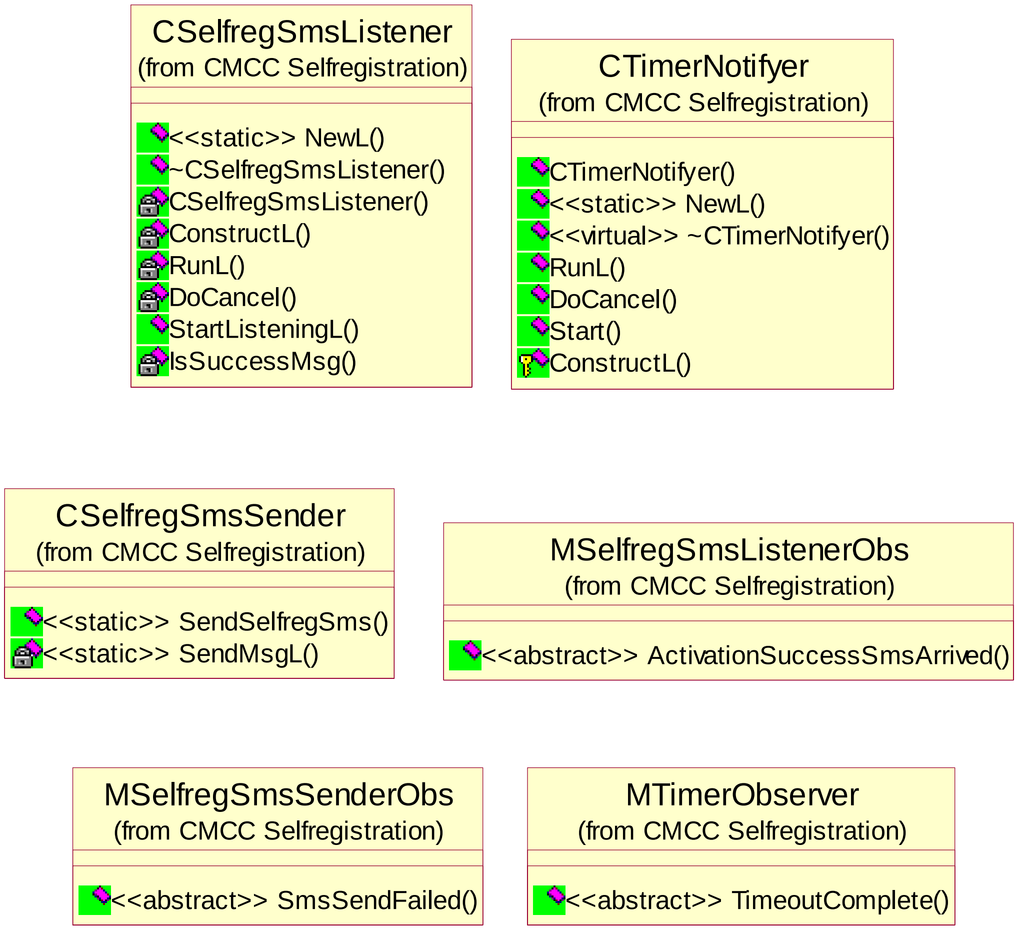
	Self-registration SMS format
11	Terminal self-registration SMS receiver number is 10654040, SMS port is16998. Self-registration SMS is free of charge.
12	Terminal self-registration SMS content and format are as follows: 1. IMEI (IMEI, same with the value in the OMA DM ./DevInfo/DeviceId); 2. Manufacturer name (Manufacturer name, must be same with the value in the OMA DM ./DevInfo/Man); It shall contain same string for the same phone model (Case sensitive). 3. Device model (Device model, must be same with the value in OMA DM ./DevInfo/Mod); It shall contain same string for the same phone model (Case sensitive). 4. Software version (Software version, must be same with the value in the ./DevDetail/SwV).
13	Self-registration SMS format is as follows: AAAAAAAAA/BBBBBBB/CCCCCCC/DDDDDDDDD Where, A represents IMEI, B manufacturer name, C device model and D software version. The sequence of the 4 pieces of information of terminal self-registration message is as follows: IMEI, manufacturer name, device model and software version. The self-registration message's length shall be less than that of an English message, i.e., 160 bytes. Examples: IMEI: 35125400847211821/Motorola/V780/Ver2.1.0.18 This SMS will be sent as Hex format, instead of ASCII format.
14	Format of the DM platform's acknowledgment message to terminal self-registration: 1. IMEI (must be same with the value in the self-registration message); 2. Correct or not: 0 represents failure and 1 success. AAAAAAAAA/B Where, A represents IMEI and B state information (0/1). Example: IMEI: 35125400847211821/0 (activation fails). IMEI: 35125400847211821/1(activation succeeds). IMEI value must be consistent with the IMEI value in the self-registration message. Otherwise, the terminal shall discard this acknowledgment message.

Appendix B – UML diagrams of application






CSelfregistrationEngine
(from CMCC Selfregistration)
<<static>> NewL() : CSelfregistrationEngine*
~CSelfregistrationEngine()
CSelfregistrationEngine()
ConstructL() : void
Start() : void
RunL() : void
DoCancel() : void
CompleteSelf() : void
ActivationSuccessSmsArrived() : void
SmsSendFailed() : void
TimeoutComplete(aError : TInt) : void


CSelfregSmsListener
(from CMCC Selfregistration)
<<static>> NewL()
~CSelfregSmsListener()
CSelfregSmsListener()
ConstructL()
RunL()
DoCancel()
StartListeningL()
IsSuccessMsg()
CTimerNotifyer
(from CMCC Selfregistration)
CTimerNotifyer()
<<static>> NewL()
<<virtual>> ~CTimerNotifyer()
RunL()
DoCancel()
Start()
ConstructL()
CSelfregSmsSender
(from CMCC Selfregistration)
<<static>> SendSelfregSms()
<<static>> SendMsgL()
MSelfregSmsListenerObs
(from CMCC Selfregistration)
<<abstract>> ActivationSuccessSmsArrived()
MSelfregSmsSenderObs
(from CMCC Selfregistration)
<<abstract>> SmsSendFailed()
MTimerObserver
(from CMCC Selfregistration)
<<abstract>> TimeoutComplete()

COS/CCS 2007-16