Web Call SDK 2.0

A dissertation submitted to the Royal Institute of Technology (KTH) in partial fulfillment of the requirements for the degree of Master of Science

SHANBO LI

Master's Thesis at ERICSSON AB
Supervisor at Ericsson: Peter Yeung
Supervisor at KTH: Mihhail Matskin, PhD
Examiner: Mihhail Matskin, PhD

TRITA xxx yyyy-nn

Abstract

TODO: write abstract here

To my parents, LI Chongzhi and WU Wei, who have guided me through life and encouraged me to follow my own path, and to my wife, MEI Dan, for being waiting for me and keeping faith in me.

Contents

Acknowledgments

Acknowledgment to Ericsson. name: Peter, Pär, Eric, Lena ... TODO: the text

Acknowledgment to KTH. name: Misha, Haseeb, Nima ... TODO: the text

Acknowledgment to Beijing Jiaotong University

Chapter 1

Background

1.1 Introduction

Voice over Internet Protocol (VoIP) enables the communications between Internet users and the endpoints in PSTN circuit switched (CS) networks. As we know, Session Initiation Protocol (SIP) is widely adopted as a signaling protocol for VoIP communications. Because of its simplicity, power and extensibility, it has also been selected by the Third Generation Partnership Project (3GPP) as a major component of IP Multimedia Subsystem (IMS) for the evolved UMTS core network.

As the world-leading supplier in telecommunications, Ericsson realizes that the position of VoIP technology in the future of telecom industry is absolutely outstanding. A project named Web Call SDK is planned by the department of Ericsson Developer Connection, to create an application that based on VoIP and make phone to phone calls.

1.2 Task

The task is to take all of the benefits of VoIP and create a powerful application for phone calls. The application should be simple, stable, reusable, extendable, integratable, easy to access and user friendly.

It should be a splendid application that supplies the function for users to make phone call anytime, anywhere and to anyone in this world.

1.3 Terminology

Java

The Java™ programming language is a popular high-level language which provides a portable feature and can be used on many different operating systems.

The source code of Java is first written in plain text files which ends with the `.java`. Then the Java source files are compiled into bytecodes which ends with `.class` by Java compiler (javac). A `.class` file is platform independent. It will be executed by the Java Virtual Machine[1].[1]

A Java application can be distributed in the format of Java ARchive (JAR) file.

Java EE

Java Platform, Enterprise Edition (Java EE) is a set of coordinated technologies that significantly reduces the cost and complexity of developing, deploying, and managing multitier, server-centric applications.[14]

Java ME

Java Platform, Micro Edition (Java ME) is a collection of technologies and specifications to create a platform that fits the requirements for mobile devices such as consumer products, embedded devices, and advanced mobile devices.[15]

MIDlet

A MIDlet is a Java application framework for the Mobile Information Device Profile (MIDP) that is typically implemented on a Java-enabled cell phone or other embedded device or emulator.

RMS

The Java ME Record Management System (RMS) provides a mechanism through which MIDlets can persistently store data and retrieve it later.[6]

JAD

The Java Application Descriptor (JAD) file, as the name implies, describes a *MIDlet* suite. The description includes the name of the MIDlet suite, the location and size of the JAR file, and the configuration and profile requirements. The file may also contain other attributes, defined by the Mobile Information Device Profile (MIDP), by the developer, or both.[13]

[1]The terms "Java Virtual Machine" and "JVM" mean a Virtual Machine for the Java platform. [1]

SSL

Secure Sockets Layer (SSL), is a cryptographic protocol which provides security and data integrity for communications over networks such as the Internet.[23]

SIP

Session Initiation Protocol (SIP) is an application-layer control (signaling) protocol for creating, modifying, and terminating sessions with one or more participants. These sessions include Internet telephone calls, multimedia distribution, and multimedia conferences.[20]

SIP is the format of control signal in VoIP. It describes the sender, receiver. A agent use SIP messages to register on a proxy, establish session or close session.

SDP

SDP is short for Session Description Protocol. It is intended for describing multimedia sessions for the purposes of session announcement, session invitation, and other forms of multimedia session initiation.[8]

A SIP message may carry a SDP message. The SDP message contains protocol version, session name, information, and most important, the connection data and media descriptions. This supplies a way to manipulate the connection of media flow.

RTP

RTP, the real-time transport protocol, provides end-to-end network transport functions suitable for applications transmitting real-time data, such as audio, video or simulation data, over multicast or unicast network services.[21]

Web Service

A web service is defined by the W3C as “a software system designed to support interoperable machine-to-machine interaction over a network”.[9]

SOAP

SOAP is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. It uses XML technologies, an extensible messaging framework containing a message construct that can be exchanged over a variety of underlying protocols.[7]

WSDL

WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information.[3]

PSTN

PSTN is short for Public Switched Telephone Network. It is the network of the world's public circuit-switched telephone networks. In another word, it is just the traditional phone network.

1.4 About

Web Call SDK *2.0* is a open source project at Ericsson Developer Connection[2] (EDC), Ericsson[3]. It is the successor project of **Web Call SDK**[2] which is developed by Yuening Chen at Ericsson AB during the year 2007 and 2008. After Yuening finished Web Call SDK, the people at EDC found it is not stable enough and many of the functions are not usable. So they decided to start a new project follow Web Call SDK to make the it stable and add some new feature to it. The new project is called **Web Call SDK *2.0***. As the main developer, I took over the new project at March 2008 and finished it at February 2009. I rewrite most code of the old project to make it stable and runnable and added some new feature to it. As a result, the **Web Call SDK *2.0*** is used as the base library of Ericsson's demo of **Using REST and Web Services to Mash Up Communications Capabilities**[5][18] at JavaOne™[4] 2009.

[2]Ericsson Developer Connection (former Ericsson Mobility World Developer Porgram) is a department of Ericsson. It helps developers to create applications that incorporate telecommunication network capabilities, such as location-based services, charging, messaging and presence, with sustainability in mind.

[3]Ericsson is a world-leading provider of telecommunications equipment and related services to mobile and fixed network operators globally.

[4]JavaOne is an annual conference (since 1996) put on by Sun Microsystems to discuss Java technologies

Chapter 2

Requirement

2.1 Programming Language

To make the application potable, **Java** is chosen as the programming language of Web Call SDK. The Java programming language is a popular high-level language which provides a portable feature and can be used on many different operating systems. For the introduction and more detail of Java please refer to 1.3.

2.2 Simple

The word "Simple" means, for community developers, the Web Call SDK should supplies a set of API that are easy to understand and convenient to use. The developers who use this API do not need much experience on java language and deep understanding of VoIP technology.

2.3 Stability

The application should be stable and has as less bugs as possible. The application should be designed for deploying on a server for long term use. The concurrent request users may more than one hundred.

2.4 Reusability

The code should be made as generic and reusable. The interface should not constrain on any specific network or service provider. It should follow a common accepted standard. Session Initiation Protocol (SIP) is a signalling protocol, which defined in RFC 3261 SIP: Session Initiation Protocol [20], widely used

for multimedia communication sessions such as voice and video calls over the Internet. It should be used as the main signal protocol of Web Call SDK.

2.5 Extendibility

The application should be able to add new feature according to customer's requirement, e.g. add video call and instant message.

2.6 Integration

The Web Call SDK should supply a web service API that can be used by other applications. This interface should contain most of the functions of Web Call SDK. And can be easily used on Web 2.0[1] mashup[2].

[1] "Web 2.0" refers to a perceived second generation of web development and design, that facilitates communication, secure information sharing, interoperability, and collaboration on the World Wide Web.[24] See also: What Is Web 2.0[16]

[2] Mashup is a Web application that combines data or functionality from two or more sources into a single integrated application.

Chapter 3

Study

3.1 VoIP Market

In the telecom market, VoIP technology has gained more and more customers. The advantage of VoIP is obviously, much cheaper fee and almost same quality as traditional telephone. Report from Infonetics indicates that, in the year 2007, the subscribers for VoIP are under 80 all around world. Most of them are in the Asia Pacific region. However by the year of 2011 the user will be 135 million, predicted by MarketResearch.com. And a UK research company Disruptive Analysis Ltd. predicts the users of mobile-VoIP will be 250 million by the year of 2012.[11]

The analyses and figures above draw a brilliant future of VoIP market.

3.1.1 VoIP Service Provider

VoIP service provider is the company which supplies the products of VoIP/PSTN gateway. Or the ones who supply the service that customers can call a PSTN phone by a VoIP phone via their service/network. There are hundreds of such companies in the world.

3.1.2 VoIP Client

A VoIP client is a common SIP client software or a IMS client software. This kind of software runs on a computer or mobile device and implements the SIP or/and IMS standard. It can work like a phone to dial or answer VoIP calls.

3.1.3 Solution Provider

A solution provider is a company that supplies both VoIP service and software client, such as Skype™[1], VoipStunt[2] and JAJAH[3]. Among them Skype is the most famous one. It has a very good quality of voice and functionality client. However, the Skype is not following the standard of SIP. So it means, only the Skype client itself can use the service of Skype. VoipStund and JAJAH supply relevant lower fee and less quality of audio.

3.2 Third Party Call Control

In the traditional telephony context, third party call control allows one entity (which we call the controller) to set up and manage a communications relationship between two or ore other parties. Third party call control (referred to as **3pcc**) is often used for operator services (where an operator creates a call that connects two participants together) and conferencing.[19]

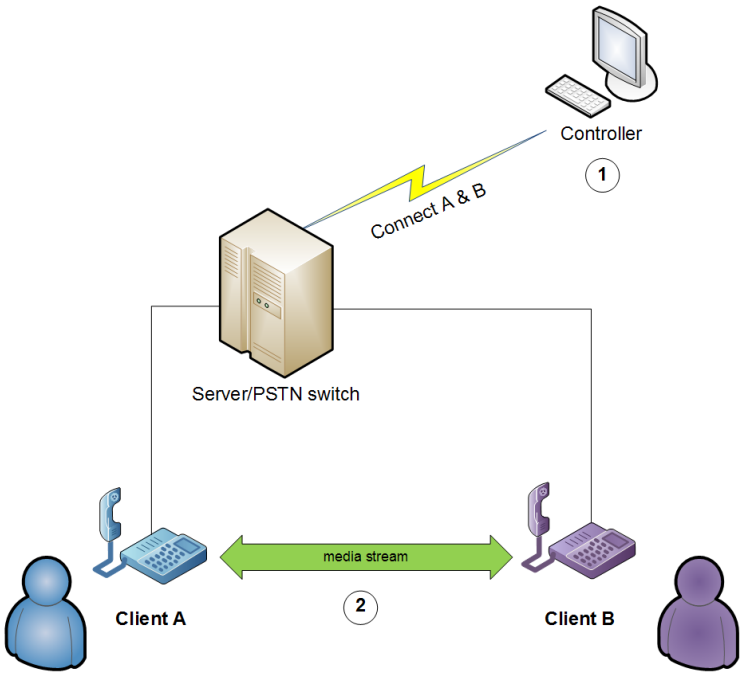

Figure 3.1. Third party call control

A general work flow of 3pcc is shown in Figure 3.1. The initial side of the phone is the *controller*. The *controller* sends a signal "connect *client A* and

[1] http://www.skype.com
[2] http://www.voipstunt.com
[3] http://www.jajah.com

client B" to the server ①. And the server establishes a call between *client A* and *B* ②.

3.3 Why Web Call SDK?

Based on IMS/SIP technology, Web Call SDK integrated call functions into Web containers. This presents a simple way to implement the communication convergence of Web, IMS/SIP network, and CS networks. It does not require the installation of the plug-in clients on the browser or other special client software as most VoIP services.

The Web Call SDK is nether a service provider or a client that described above. It is more like a controller which acts as a initial side in third party call control. That is, it support all standard client and service provider. Another advantage of Web Call SDK is that The desktop view, mobile browser view and JavaME client all share a same database. The user can access a same contact book and use a same service account from different platform. None of the solution provider or client have the same function.

Chapter 4

Solution

There are two kind of connection solution of the core of Web Call, the Relay Call and Third Party Call. The different between them is the way they handle media stream.

The Relay Call Controller works as a back-to-back agent and forwards media streams, while Third Party Call Controller only establishes connections by sending out SIP messages and it does not handles any streams itself.

4.1 Relay Call

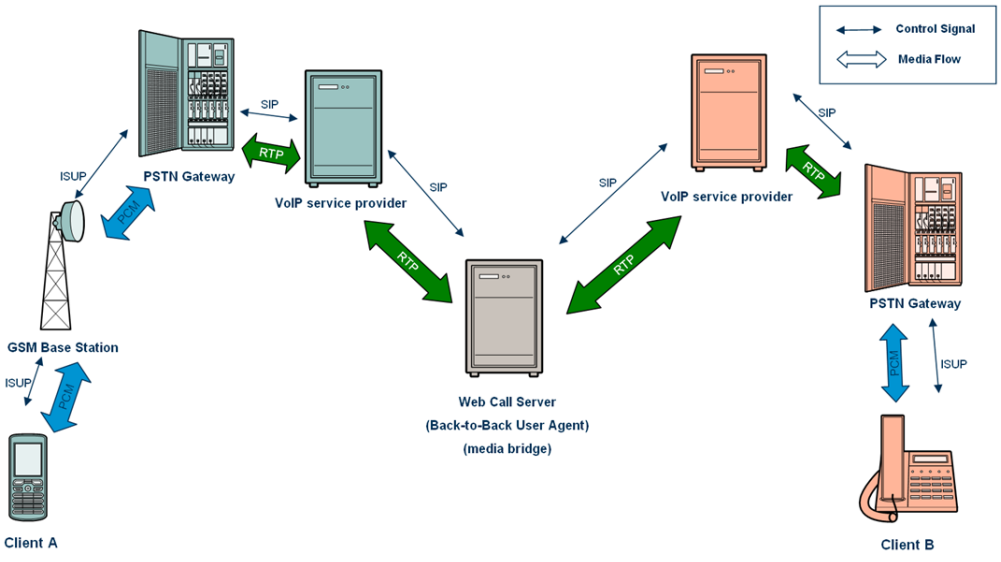

Figure 4.1. The signal and media flow of Relay Call

The signal and media flow of Relay Call is shown in Figure 4.1. In this

scenario, the Web Call Example Application acts as a back-to-back user agent. It sets up the connection and forwards the media stream. It can be seen from the picture that both signal and media are handled by Web Call Server. When it starts, it try to call client A. After it establishes a session with client A, it will try to call client B and also establish a session with client B. After that, it will work as a media stream bridge and forward media stream from client A to B, as well as from client B to A.

For a detail description and mechanism of Relay Call, please refer to the master thesis of **Web Call SDK** by *Yuening Chen*[2].

4.2 Problem of Relay Call

4.2.1 The Load on Controller

The controller here acts as a back to back user agent (B2BUA). It receives the RTP flow from one client and transfers it to another client. It does the same in the opposite direction. That means all of the RTP traffic will go through the controller. So the load on controller will be heavier as with the number of concurrent users increases. A powerful host is needed to handle the RTP flows. However the design goal of Web Call SDK was simply a tool kit that can easily be integrated into a web site. Unfortunately, this current mechanism of session control will decrease the performance of whole web site.

4.2.2 The Latency of Audio

As can be seen from the session flow, the RTP goes from one client to the controller via a SIP provider and then the controller transfers the RTP to another client via the SIP provider again. This means that each direction of RTP will go through SIP provider twice. So the latency of the phone call via the SIP provider will be double that of normal calls. The latency of phone-to-phone call via Web Call SDK test turned to be more then 2 seconds. It is quite unacceptable.

4.2.3 Lack of Reliability

Since all of RTP flows go through the controller, thus if the controller crashes, all of calls will be immediately interrupted.

4.3 Third Party Call

In the traditional telephony context, third party call control allows one entity (which we call the controller) to set up and manage a communications relationship between two or more other parties. Third Party call control (referred as 3pcc) is often used for operator services (where an operator creates a call that connects

two participants together) and for conferencing. The signal and media flow in third party call is show in Figure 4.2 The advantage of third party call in Web Call is that the controller only need to handle message transfer and leaves the media flow for the ISP.

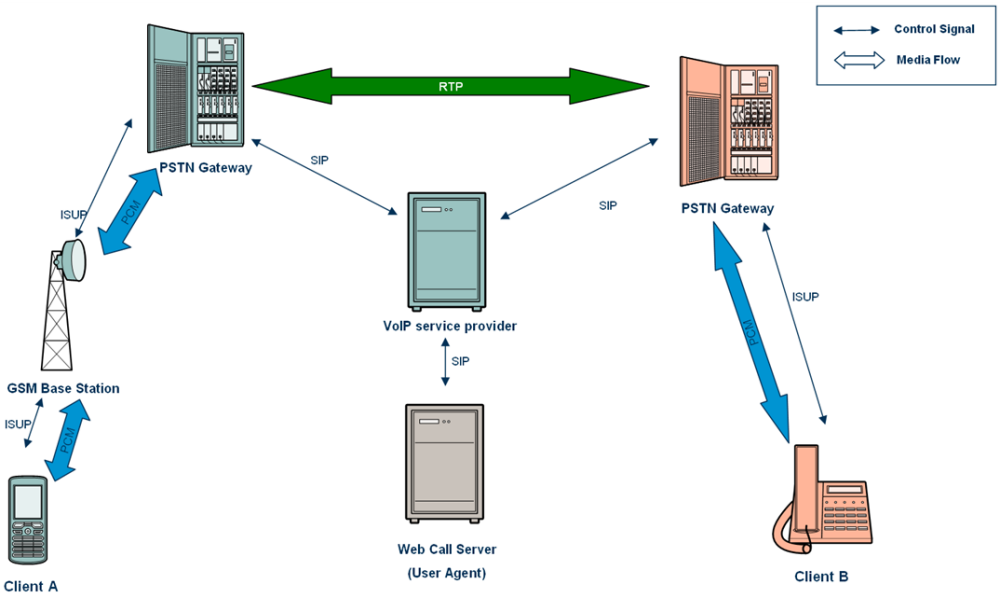

Figure 4.2. The signal and media flow of Third Party Call

4.3.1 Call Transfer

The call transfer implementation use a `REFER` method which defined in *The Session Initiation Protocol (SIP) Refer Method* (RFC 3515)[22]. "The `REFER` method indicates that the recipient (identified by the Request-URI) should contact a third party using the contact information provided in the request"[22].

The Call flow is shown in Figure 4.3. The controller first sends an `INVITE` to client A (1). This invite is just a normal invite. A's phone rings and answers. This results in a `200 OK` (2). The controller then answer client A an `ACK` (3). Follow that, the controller send out a `REFER refer-to`: Client B (4), which means controller wish client A to make a phone call to client B. The client A understands that and returns a `200 OK` to controller (5). Then client A sends an `INVITE referred-By`: C to client B (6). Client A and Client B can establish a session according the `INVITE` from A to B. The `BYE` (8) and `200 OK` (9) means the controller cut the media stream between itself and client A.

To accomplish the whole call flow, client A must support RFC3515.

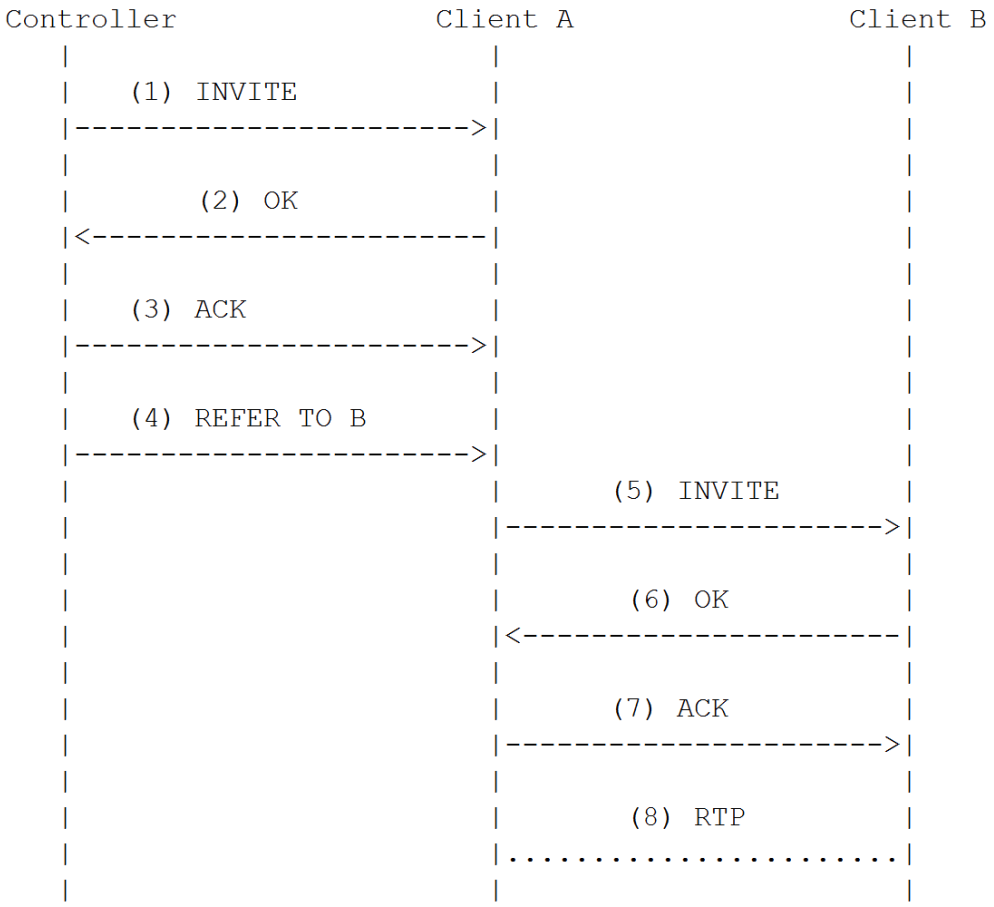

Figure 4.3. The signal and media flow of Call Transfer

4.3.2 SDP Swap

This implementation of third party call control swaps SDP from two clients. The prototype of SDP swap comes from *Best Current Practices for Third Party Call Control (3pcc) in the Session Initiation Protocol (SIP)* (RFC 3725). However, the call flow in RFC 3725 is quite complicated. The concept of **SDP Swep** is that controller calls to client and swaps two SDP which got from clients. These two clients seem to call controller but the media stream is connected directly between them.

The call flow of SDP swap implementation of third party call control is shown in Figure 4.4. Controller first call client A. This `INVITE` has no session description. Client A's phone rings, and client A answers. It results a `200 OK` (3) that contains an offer which contains client A's SDP [20]. Meanwhile, the controller does the same to client B (2) and gets client B's offer (4) which contains client B's SDP. So far, controller has both client A and client B's SDP. It just simply swaps these two SDPs, then `ACK` client B with client A's SDP (5) and `ACK` client A with client

B's SDP (6). So client A gets client B's SDP by (6) and client B gets client A's SDP by (5). Therefore, media flows between client A and client B (7).

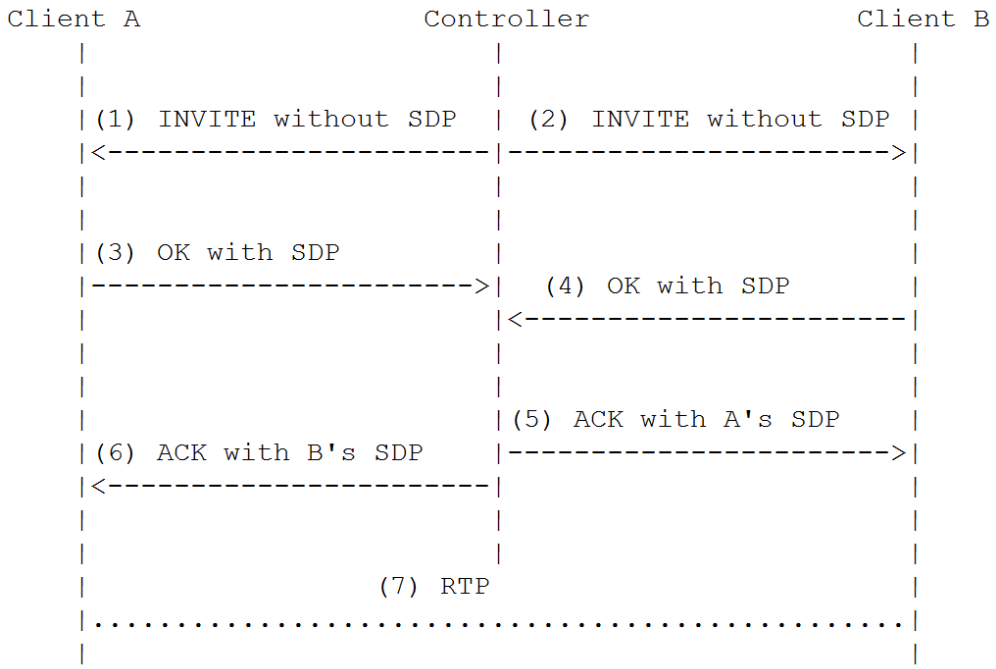

Figure 4.4. The signal and media flow of SDP Swap

4.3.3 Re-invite

A `Re-INVITE` is used to change the session parameters of an existing or pending call. It uses the same Call-ID, but the CSeq is incremented because it is a new request. The Re-invite implementation send two client's SDP to each other in the Re-invite process.

The call flow is shown in Figure 4.5. From the SIP message (1) to (6), the controller uses a three way handshake to establish connections with client A and client B. In (3) and (4) the controller gets client A and client B's SDP which are going to be used in the Re-invite phase in (7) and (8). The controller sends a `Re-INVITE` to A with B's SDP (7), which indicates the controller changes its media port to B's. At the moment, controller also send a `Re-INVITE` to client B with client A's SDP, which indicates the controller change its media port to A's. Both client A and client B gets each other's SDP, thus, a media stream could be established between A and B (13).

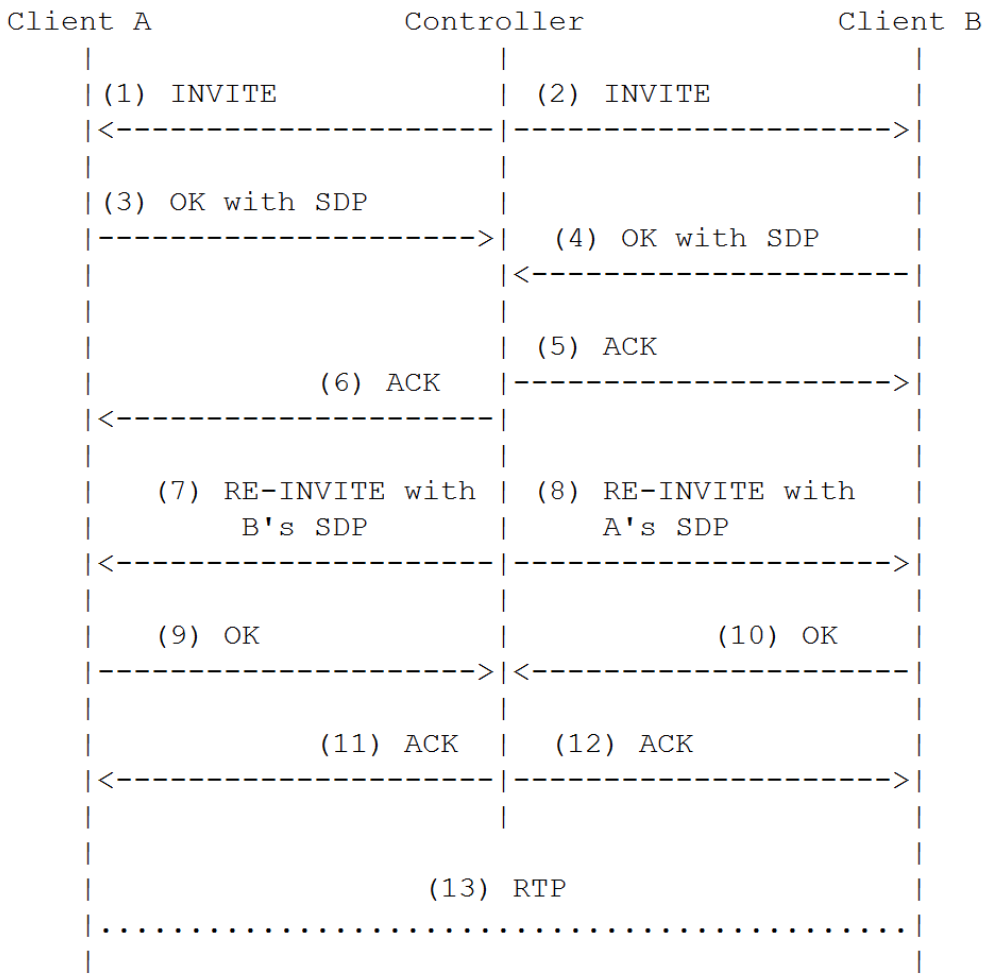

Figure 4.5. The signal and media flow of SDP Swap

4.3.4 Web Client

Most of the VoIP service provider supply a way of make phone to phone call via their web site. To establish the call, user have to login to the web site and fill in the caller number and callee number. The web client way of third party call use a project called **scallope**[12]. It is a application which acts as a web client and login to sip provider's web page and to make VoIP calls.

Scallope is a Java API which based on `apache common http` API. All user need to do is just pass the user name, password, caller number and callee number to the API. The Scallope will access VoIP service provider's web page via http protocol and establish a call session for user.

This kind of 3pcc makes it possible to start a phone to phone call with out a web browser. For the aspect of control signal and media flow, everything is within provider's network. So the quality of audio is quite close to PSTN and

very much acceptable.

4.4 Conclusion

It can be seen from Table 4.1, when talk about latency, the best call method is the PSTN (the traditional way). But it costs a lot for international sessions. The Relay call can be used on any server and client with a low fee. But the latency is long and sometimes unacceptable. The relay call method brings media traffic to the controller so it not possible to have it on a small or personal server. The third party call is not support by all of the services. We have developed four different solutions on third party call. Each of them has its own Cons and Pros. It is recommended that when make an international or long distance call, the web client method should be choose. If it is not possible to use web client method, users can try the other third party call implementations. If none of them works, user can choose either the PSTN which supply a good quality and high cost or the relay call which supply a poor quality but with low price.

<table>
<tr><td rowspan="3"></td><td rowspan="3">PSTN</td><td colspan="5">VoIP</td></tr>
<tr><td rowspan="2">Relay Call</td><td colspan="4">Third Party Call</td></tr>
<tr><td>Call Transfer</td><td>SDP Swap</td><td>Re-Invite</td><td>Web Client</td></tr>
<tr><td rowspan="2">service provider</td><td rowspan="2">All PSTN switch</td><td rowspan="2">support by all VoIP service providers</td><td colspan="4">Not support by all VoIP service providers</td></tr>
<tr><td>The ones who support REFER method</td><td>The ones who do not filter SIP message which doesn't carry SDP</td><td>The ones who support Re-Invite message</td><td>The ones who supply web site call</td></tr>
<tr><td rowspan="2">Client</td><td rowspan="2">Traditional telephone or mobile phone</td><td rowspan="2">All clients (traditional phone, mobile phone, software-client)</td><td colspan="3">Traditional phone, mobile phone, software-client, under particular requirements of software-client or PSTN gateway</td><td rowspan="2">All clients (traditional phone, mobile phone, software-client)</td></tr>
<tr><td>Client need support REFER method</td><td>Client need implement RFC 3725</td><td>Client need support Re-Invite</td></tr>
<tr><td>Media Stream</td><td>In PSTN network</td><td>Internet and/or PSTN, need to be handled on controller</td><td colspan="4">Internet and/or PSTN</td></tr>
<tr><td>Latency</td><td>Very Short / QoS guarantee</td><td>long</td><td colspan="4">Short, acceptable. But no QoS guarantee</td></tr>
<tr><td>Cost</td><td>Expensive (especially for international call)</td><td colspan="5">cheap</td></tr>
</table>

Table 4.1. Comparison of Call Method

Chapter 5

Application Overview

5.1 Use Cases

TODO: write some use cases as well as figures here

5.2 Architecture

This chapter presents a brief solution description of Web Call SDK. The design architecture is shown in Figure 5.1.

Web Call SDK contains three components which are SIP Call Component, Web Application (include web service interface) and Java ME client. The three components are shown with dark blue back ground in Figure 5.1.

5.2.1 SIP Call Component

The SIP Call Component is the core of Web Call SDK. It implemented 1 kind of relay call and four kind of third party call. It can be also used as a stand alone VoIP high-level API.

The details about SIP Call Component will be described in Chapter 6.

5.2.2 Web Application

The Web Application is built on the architecture of Mobile Front Controller (MFC). SIP Call Component integrated into the web server as Java EE components: servlet, and web service. The Mobile Front Controller is used for detecting and selecting views, i.e. applications with views for desktop and mobile browsers. So Web Call Example Application supplies both XHTML view which used by desktop browser and XHTML-MP view which used by mobile browser.

The details about Web Application will be described in Chapter 7.

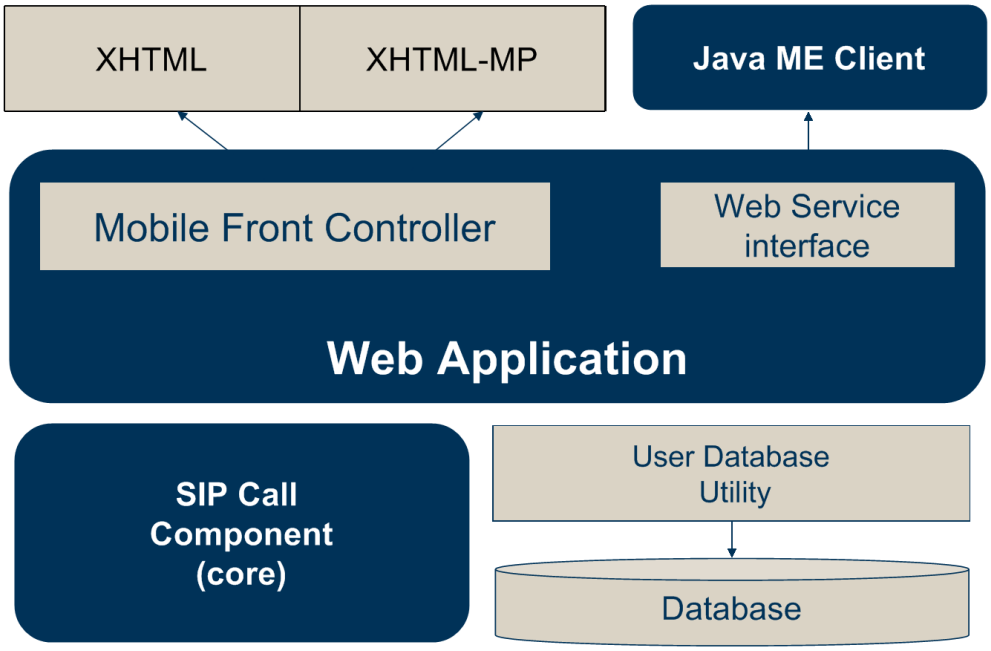

Figure 5.1. The Architecture of Web Call SDK

5.2.3 Web Service Interface

The web service interface in web application supplies a common interface for using sip call function. It uses a same database as MFC based Web Application.

The details about Web Service Interface will be described in Chapter 8.

5.2.4 Java ME client

Java ME client is a client of web service interface. It not only implement all of the client side function of web service interface, but also include some convenient function such as read phone contact book and synchronize with server contact book.

The details about Java ME client will be described in Chapter 9.

Chapter 6

SIP Call Component

6.1 Five layers architecture

A five layers architecture has to be introduced before illustrate the Third Party Call Controller. The five layers architecture is shown in Figure 6.1. From bottom to top, the five layers are Protocol stack, Abstraction layer, Implementation layer, OOP use-case based API layer and JavaBean for synchronized communication.

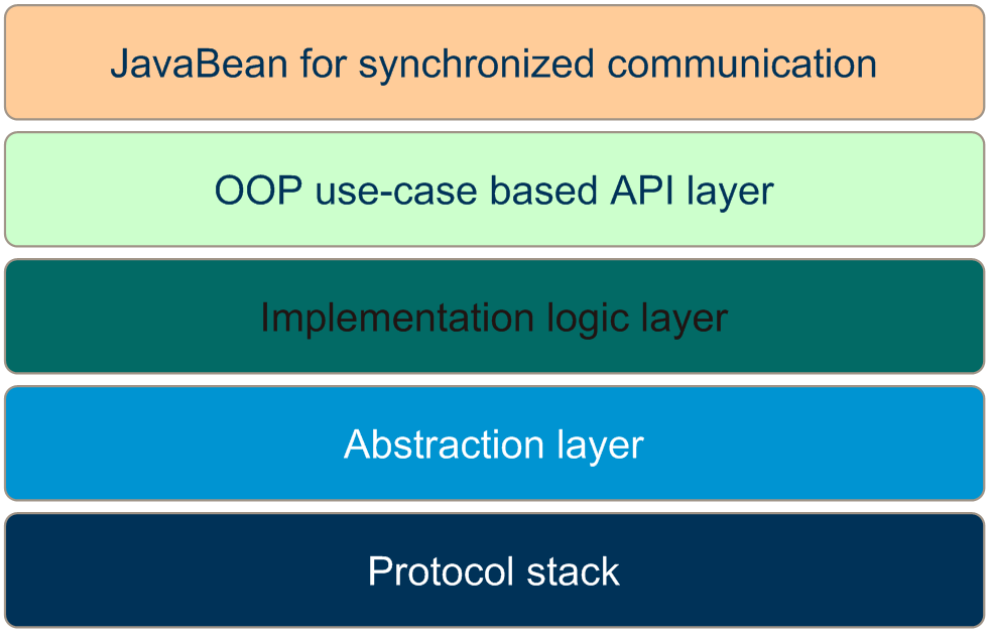

Figure 6.1. Five layers architecture

6.1.1 Protocol stack

The protocol stack is the low level API for the project. There could be several implementations of the protocol stack.

6.1.2 Abstraction layer

This layer is used for abstracting low level protocol stack. So the project can be easily migrated from one protocol stack to another without modifying the logic layer.

6.1.3 Implementation logic layer

This layer contains the main logic of the project.

6.1.4 OOP use-case Based API Layer

This layer is used for exposing public API for end user. Design Patterns such as Abstract Factory Pattern can be used here, especially for java SE.

6.1.5 JavaBean for synchronized communication

Components in Java are called beans. This layer contains The JavaBean which special designed for synchronized communication has two pairs of constructor and business logic group. The constructor which has system parameters works with business logic which doesn't have system parameters. The constructor which doesn¡¯t have system parameters works with business logic which has system parameters. In addition, this kind of JavaBean only has method of `getState()` and no method of `setState()`.

6.2 Architecture of SIP Call Component

The SIP Call Component is the core of Web Call Example Application. The design architecture of it is shown in Figure 6.2.

TODO: write something here

6.3 Hierarchy of Call Controller

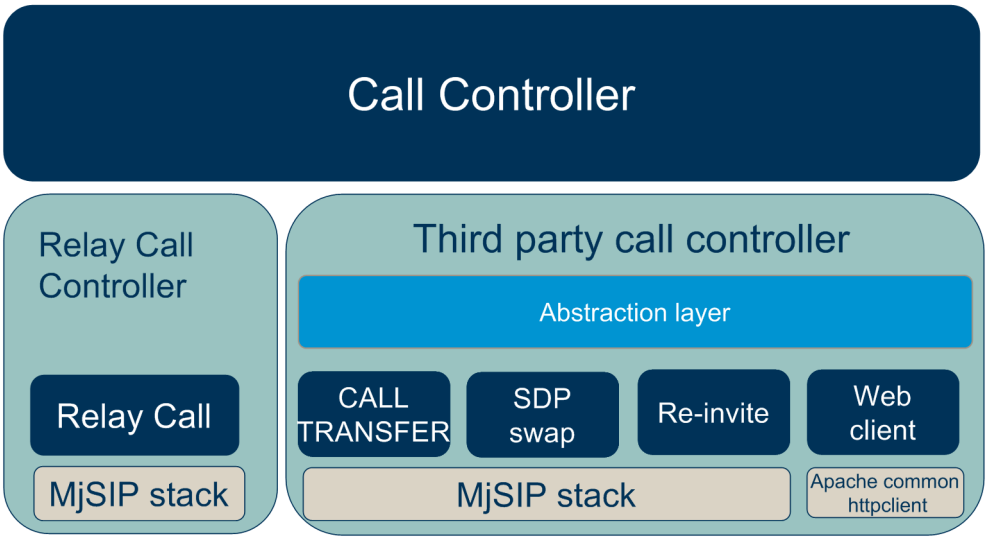

Figure 6.2. The Architecture of SIP Call Component

Chapter 7

Web Application

7.1 Overview

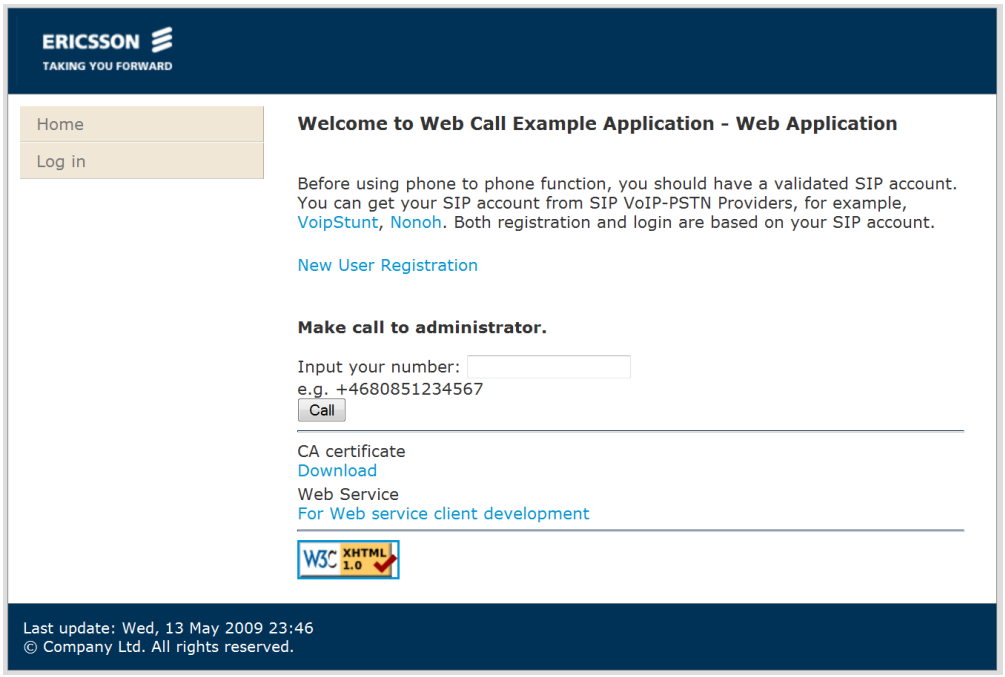

Figure 7.1. Welcome page of desktop browser view of web application

The web application is the main portal of the whole application. It is based on Mobile Front Controller. The role of web application in the whole application is shown in Figure 5.1. Users can use the web application to manage their accounts and VoIP calls. Administrators can use web applications to manage users. The web application can be packaged into a war file and easily to deploy. It only

needs a sevlet container, like Tomcat, to run. It supplies two kind of views one is for desktop browser and another is for mobile browser. Beside these two views, the web application also contains a web service interface. This chapter will focus on two browser view. The web service interface will be introduced with detail in chapter 8.

A welcome screen of desktop browser view of Web Application is shown in Figure 7.1.

7.2 Architecture of Mobile Front Controller 3

Mobile Front Controller (MFC) is a light-weight Java EE web application framework for creating web applications for web browsing and mobile browsing. It was developed by Peter Yeung and Pär Johansson from Ericsson Developer Connection, Ericsson[25]. The mobile front controller uses a sevlet to handle http request, and redirect request to different kind of view. All views share a same logic.

The following is an abstract from the documentation of Mobile Front Controller SDK. *"Mobile Front Controller (MFC) is a light-weight Java EE web application framework based on design patterns. MFC is used for creating internet web and mobile applications, i.e. applications with views for web browsing and mobile browsing that share UI logic."*[25]

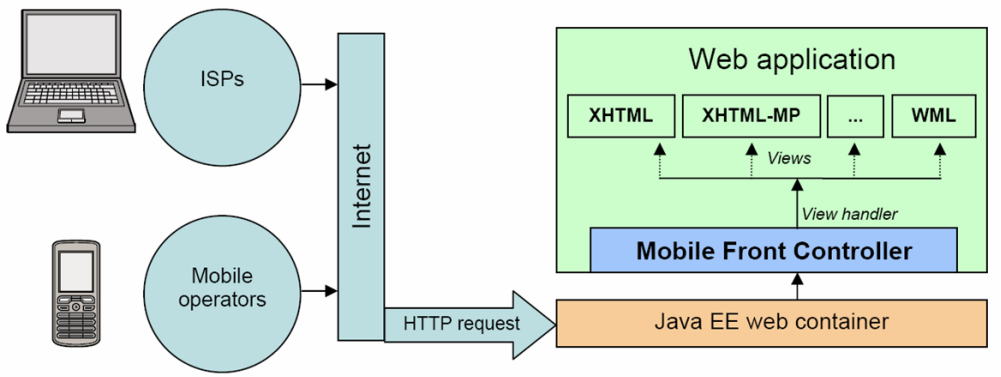

Figure 7.2. An overview of Mobile Front Controller used by a web application on a Java EE web container. (Figure taken from Mobile Front Controller Developer's guide for software version 3.1[10])

7.3 Site structure

7.4 Desktop browser view

7.5 Mobile browser view

7.6 User action

7.7 Administrator action

7.8 Validation mechanism

7.8.1 Page level

7.8.2 Server level

7.9 Session control

7.10 Ajax in web application

7.11 Java ME helper

7.12 Database

TODO: remember to write the first user will be administrator.

7.12.1 Design of database

7.12.2 User database utility

7.13 Security

7.13.1 Certificate

7.13.2 Security constraint

7.13.3 Authorization

Chapter 8

Web Service Interface

8.1 Metro

8.2 SOAP web service

8.3 Synchronize

Chapter 9

Java ME Client

The Java ME Client is a client of web service interface which described in chapter 8. It implemented all of the interfaces of web service of Web Call SDK. Besides the web service interface, Java ME Client also provide some convenient function such as load contact book from mobile phone and synchronize it with Web Call SDK server.

9.1 Architecture

The architecture of Java ME web service client is shown in Figure 9.1. It can be seen from the diagram that there are three layers in the Java ME client. They are, from bottom to top, Java ME API, Business Logic and User Interface.

The Java ME API layer is the standard API set for Java ME platform. To meet the requirement of installation, the mobile device mast have a support of `JSR 172` (J2ME™ Web Services Specification)[4] and an optional support of `JSR 75` (PDA Optional Packages for the J2ME Platform)[17].

In business logic layer, there are several components that control the logic. A very important one, which is also the core of Java ME client, is the `Web Call Client`. The Web Call Client bases on `Web Call Client Stub` which is the client stub of web service interface of Web Call Example Application. The detail of Web Call Client and Web Service Client Stub will be discussed separately in section 9.5 and 9.2. The Web Call Client also uses a component named `Record Store Manager` which will be described in detail in section 9.3. The last component is `PIM Contact Helper`. It is a utility which helps to load contact book from mobile device.

In user interface (UI) layer, there are two implementation of UIs. The details of user interface will be shown in section 9.6.

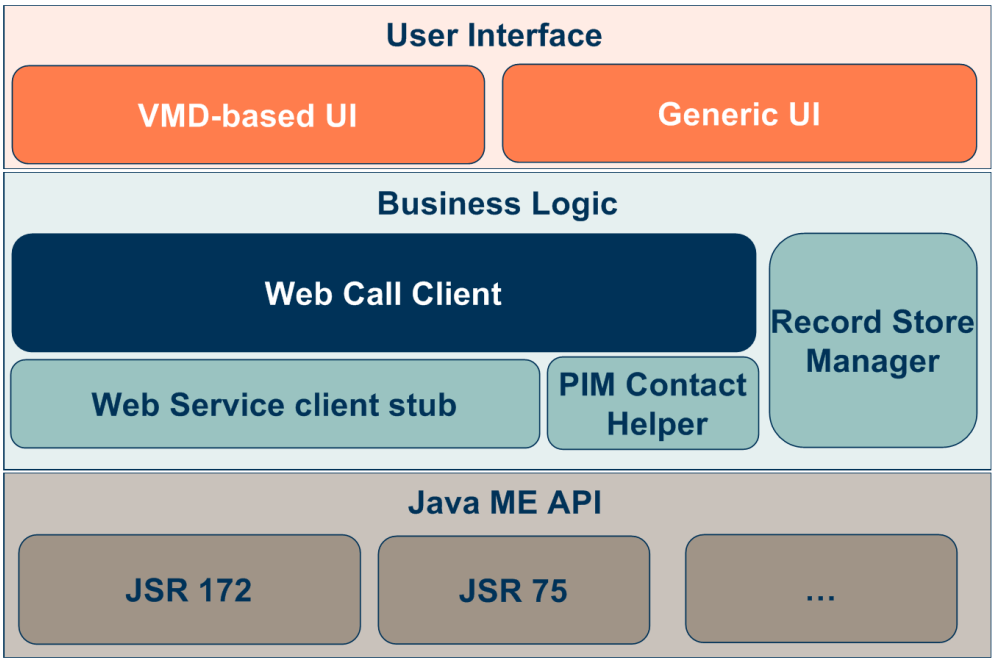

Figure 9.1. The Architecture of Java ME Client

9.2 Web Service Client Stub

The web service client stub is a stub of web service interface which described in chapter 8. It is generated by a wizard from NetBeans IDE. NetBeans is a open-source and free IDE sponsored by Sun Microsystems. The generation wizard is shown in Figure 9.2. The client stub is based on JSR 172, so only the handset which supports `JSR 172` can run the Java ME web service client.

Once the wizard is finished, NetBeans will automatically generate some classes that wrap the stub and make the operation of web service easily. And it communicates with web service server via SOAP protocol. The web service client stub is designed to be separately from the core logic of Java ME client (Web Service Client). This makes the lower stack of web service exchangeable. As just mentioned above, the client must be installed in a `JSR 172` enabler. The project team is planning to define a RESTful web service and a RESTful web service client to replace current SOAP based web service client stub. Since The RESTful web service only need a HTTP connection which is widely supported by modern mobile phones, the Java ME client will be installed on more devices by then.

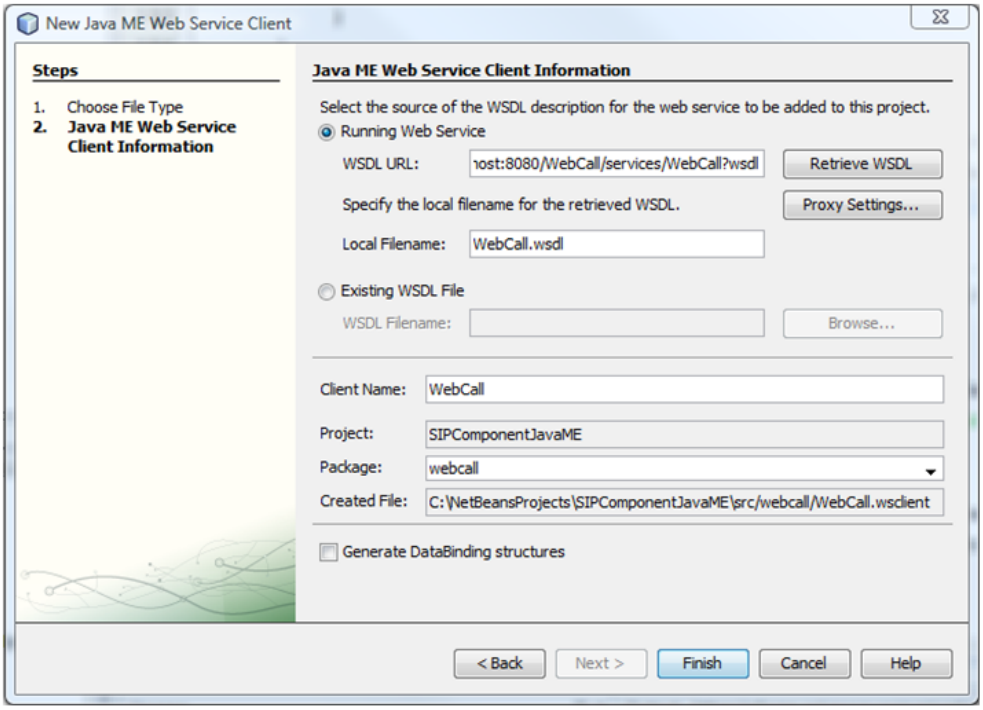

Figure 9.2. Netbeans Java ME Web Service Client Wizard

9.3 Record Store Manager

Record Store Manager is used for reading and writing the data from/to Java ME *record management system* (RMS). It extends a abstract class which named `AbstractRecordStoreManager` which is a stand alone component which supplies a very convenient way to interactive with RMS. The `AbstractRecordStoreManager` wraps the action of saving and loading data to/from RMS. Two methods `setData()` and `getData()` handles everything. The `AbstractRecordStoreManager` can be not only used in Java ME Client of Web Call Example Application, but also other Java ME applications. A class hierarchy diagram of `RecordStoreManager` and `AbstractRecordStoreManager` is shown in Figure 9.3.

9.4 PIM Contact Helper

PIM Contact Helper provides access to contact list of *Personal Information Management* (PIM) data on J2ME devices. The PIM API is designed for a widely use and try to compatible with most of Java ME enabler. But it is lack of user friendly. The PIM Contact Helper wraps some APIs of JSR 75 and makes it very easy to load contact lists from mobile phone. Like Record Store Manager, PIM Contact Helper is also a stand alone component. It can also be used in other Java

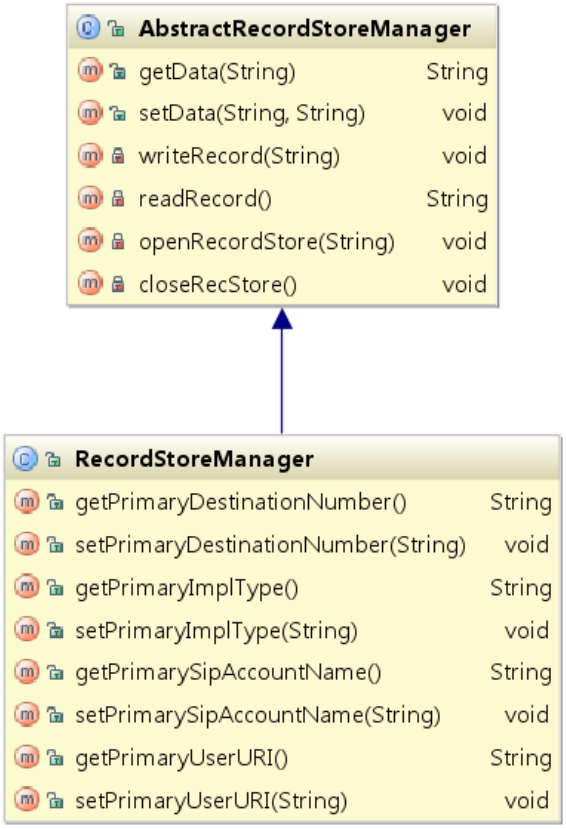

Figure 9.3. The Class Diagram of Record Store Manager

ME applications. To use it, the mobile device must support `JSR 75`[17]. When first the time use it, there will be pop up option window to ask if user want this application to read contacts list as shown in Figure 9.4. As long as user approved, a phone contacts list will be shown on screen as shown in Figure 9.5.

9.5 Web Call Client

The Web Call Client component bases on the Web Service Client Stub. As described in section 9.2, The Web Service Client Stub implements only the client interface. And the `Web Call Client` manages the business logics, e.g. the load of last used configuration, the recent call list and logic of synchronization. The Web Call Client can be treated like a adapter between UI and Web Service Client Stub. It collects phone call information from UI and set them as parameters of Web Service Client Stub.

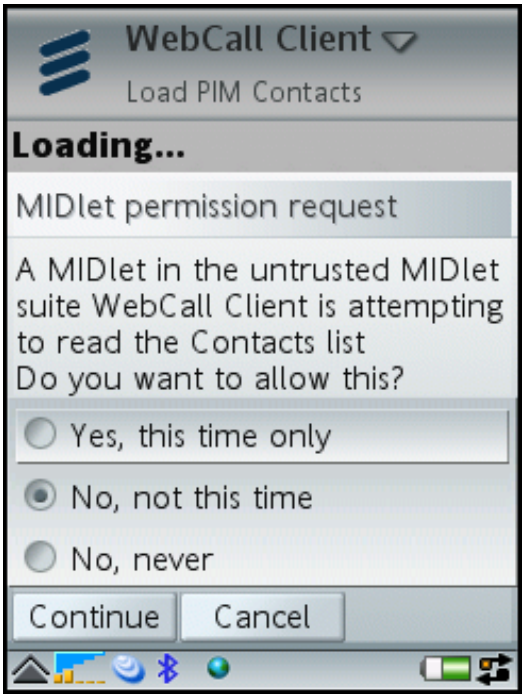

Figure 9.4. Load PIM Contacts

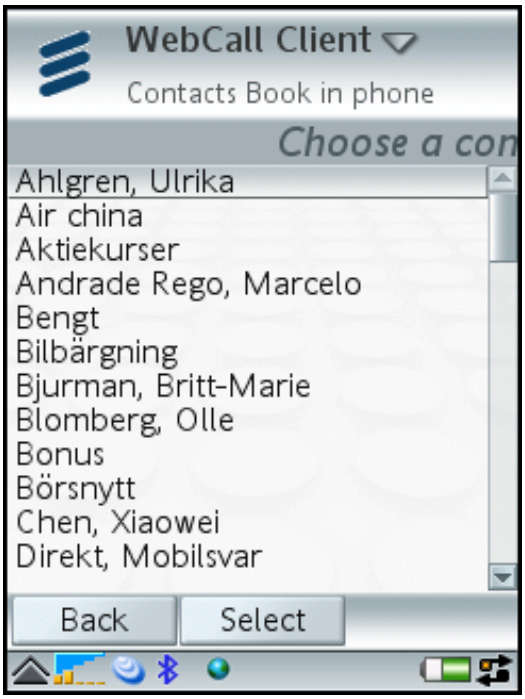

Figure 9.5. PIM Contacts List

9.6 User Interface

9.6.1 VoIP Call Form

A VoIP call form is shown in Figure 9.6. On top of that form, there are two input fields, *your phone number* and *destination phone number*. Under that, where are two drop down lists. One is *VoIP provider account*, and another is *call method*. The contents of these two lists are fetched from web service interface. The VoIP provider is used to choose a VoIP account that user wishes to use. Users can also change the content of VoIP provider drop down lists via the desktop browser view or the mobile browser view. See how to configure the VoIP provider account, please refer to section **TODO: section number** . Below that, there is a drop down list to choose call methods. There are five different methods to use. They have similar effects but not the same with of establishing connection. The differences of call method is talked in section **TODO: section number**. A screen shot of call method drop down list is shown in Figure 9.7. The left bottom of this page is the "Call" button which will connect the phone call when use click it. Next to the call button is the "Exit" button.

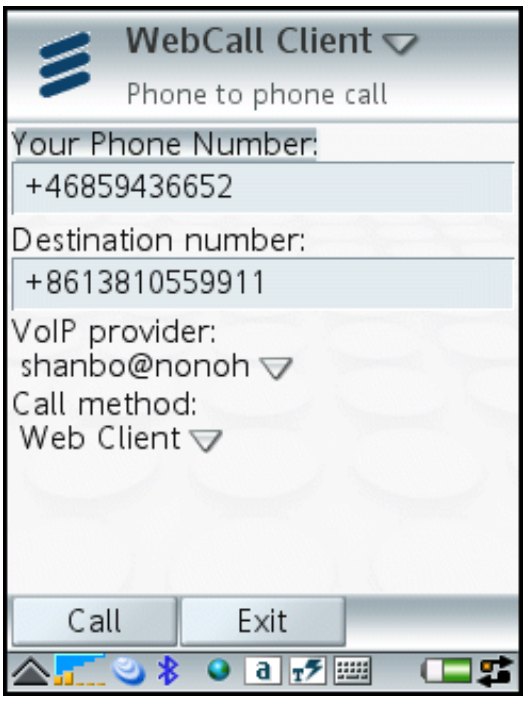

Figure 9.6. VoIP call form of Java ME Client

The Java ME client use the RMS (Record Management System) to store the last call information include caller, callee, VoIP provider and call method. When the Java ME client starts again, it will read the record from RMS and

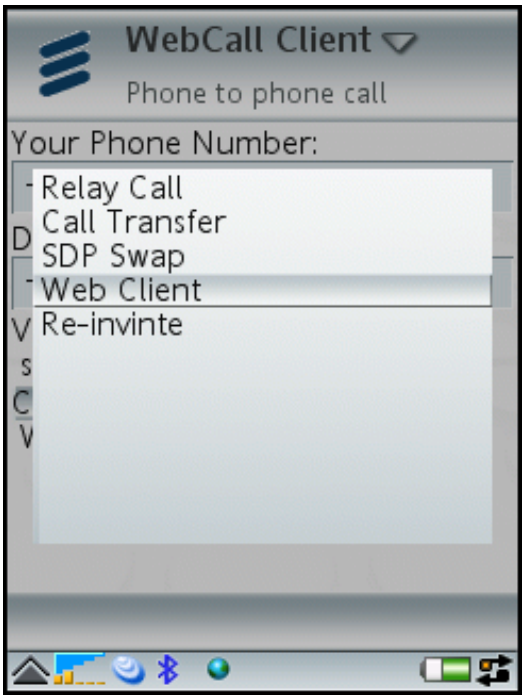

Figure 9.7. Call method drop down list

automatically set call method as the same as last time used. The interaction with RMS is handled by Record Store Manager which is described in section 9.3.

On the VoIP call form menu, there are five screen menu items, PIM Contacts, My phone number, Contacts, Recent Call and Sync which is shown in in Figure 9.8. The functionalities of five menu items as well as phone call will be introduced within follow subsections.

9.6.2 PIM Contacts

PIM contacts supplies feature of reading contacts from Personal Information Management (PIM) data from cell phone. This function is implemented by the component of PIM Contact Helper which is described in section 9.4. In the view of "Contacts Book in phone" there is also another command to save your contacts in contact book of Java ME client. As long as user uses the synchronize function, the new contact will also be stored in the remote database of web application.

9.6.3 My Phone Number

My phone number is the phone number you wish to call from. It could your current phone or a landline phone. You can have more than one phone number saved in the database of server side. Click one of "my phone number", the number will be set as your phone number in the phone to phone call form.

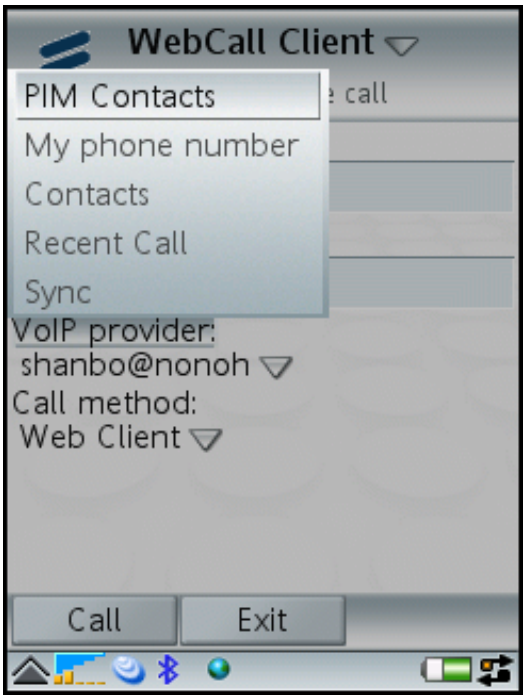

Figure 9.8. Screen menu items of VoIP call form

9.6.4 Contacts

The contacts here refer to the contacts book in the web application. It is not same as the PIM contact list in phone memory. However, you can save contacts from contacts book in phone to contacts book in web application by the function mentioned in section 9.6.2. The number of selected contact will be set as destination number. The view of Contact list is shown in Figure 9.9.

9.6.5 Synchronize

The synchronize function in phone call form is used to synchronize VoIP provider account and contacts. If same name with different phone number happens, the number in client will be kept. The synchronize function is shown in Figure 9.10.

9.6.6 Phone Call

After the user setup the phone number, destination number, VoIP provider account and Call method, just simply click “Call” button then a session will be established between the user and his friend.

The phone call function in Java ME client is easy and convenient. Every time it restarts, the configuration will be the same as last time he used. It is especially

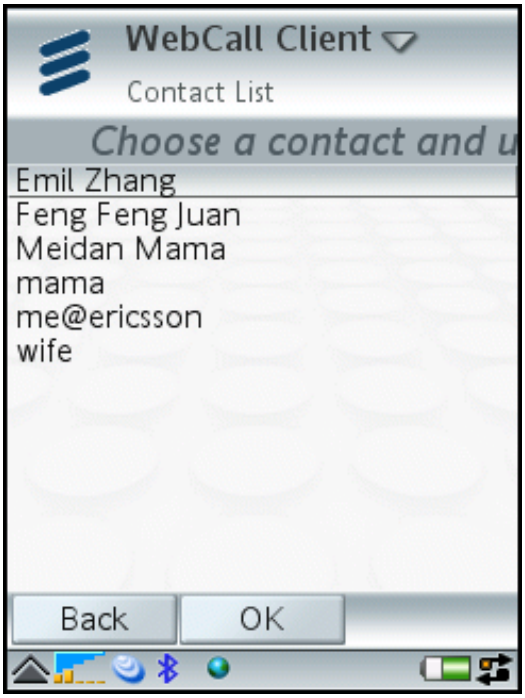

Figure 9.9. Contact list of VoIP call form

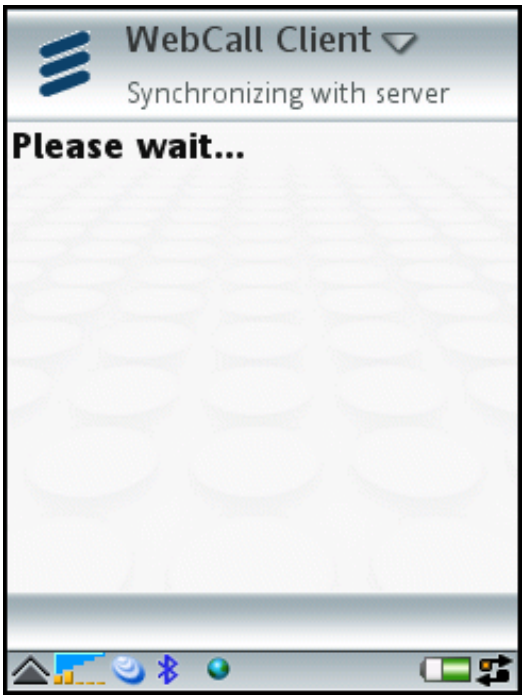

Figure 9.10. Synchronize with server

useful when the user always calls the same number. The phone call function is shown in Figure 9.11.

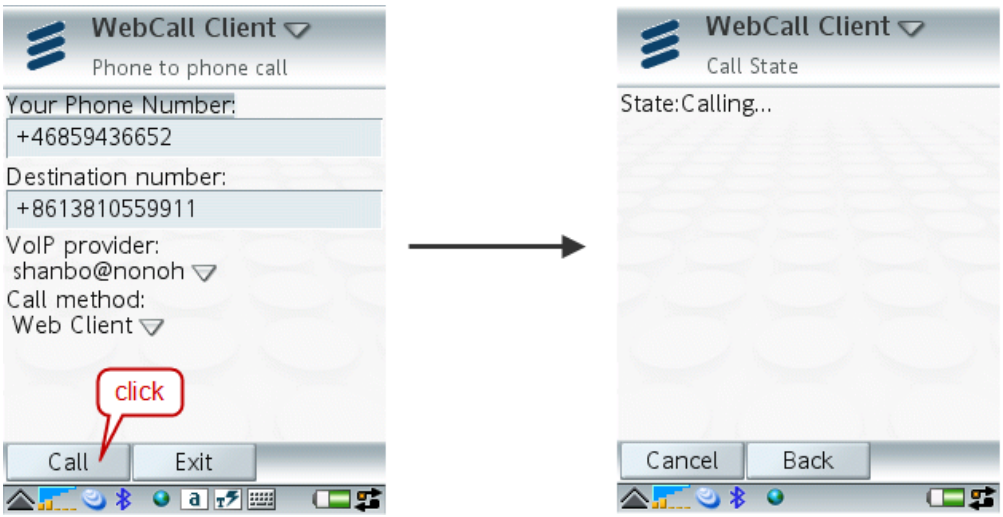

Figure 9.11. Phone call (via VoIP technology) of Java ME client

9.7 Two implementations of UI

9.7.1 VMD-based UI

This Java ME client is first developed with Visual Mobile Designer. The Visual Mobile Designer (VMD) is a graphical interface within NetBeans Mobility that enables you to design mobile applications using drag and drop components. When compile and deploy the Java ME application, The application must include a NetBeans VMD library.

The VMD-based working flow is shown in Figure 9.12. For a larger image, please refer to the *original version*[1] of working flow on web.

9.7.2 Generic UI

To avoid using the VMD library, a generic UI is introduced. It doesn't use any particular class from NetBeans. Change `WaitScreen` to `Form`, change `SimpleCancellableTask` to normal method. There are three sub packages in package `sip.components.me.ui`, `nb`, `nb2` and `general`. The package `nb` contains the VMD-based UI. The package `nb2` is a UI which doesn't contain any NetBeans library. The UI in package `general` is the code copy from nb2 and refractored according the package name and Midlet name. All three UIs can be treaded as

[1] http://shanbohomepage.googlecode.com/svn/trunk/master_thesis/chap09/resources/java_me_working_flow.png

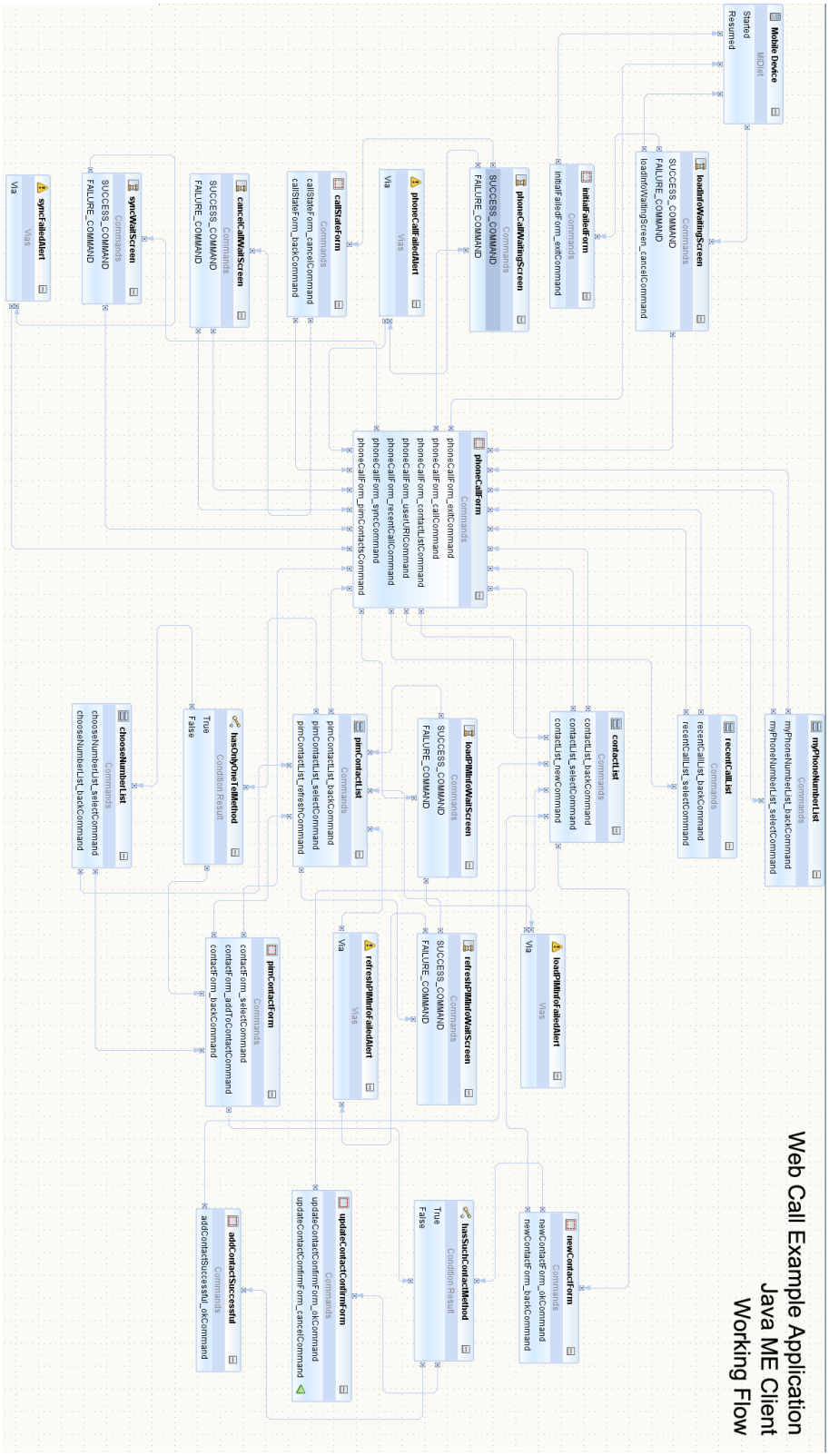

Figure 9.12. Java ME Client Working Flow

Midlets. They invoke the same logic in Web Call Client and PIM Contact Helper. By default only the UI in general will be compiled.

9.8 Installation of Java ME Client

The installation of Java ME Client could be down via mobile browser. After login to mobile view of web application, go to the user profile page, users will see "Download your JAD file", as shown in Figure 9.13. Click the link to download and install the Java ME client.

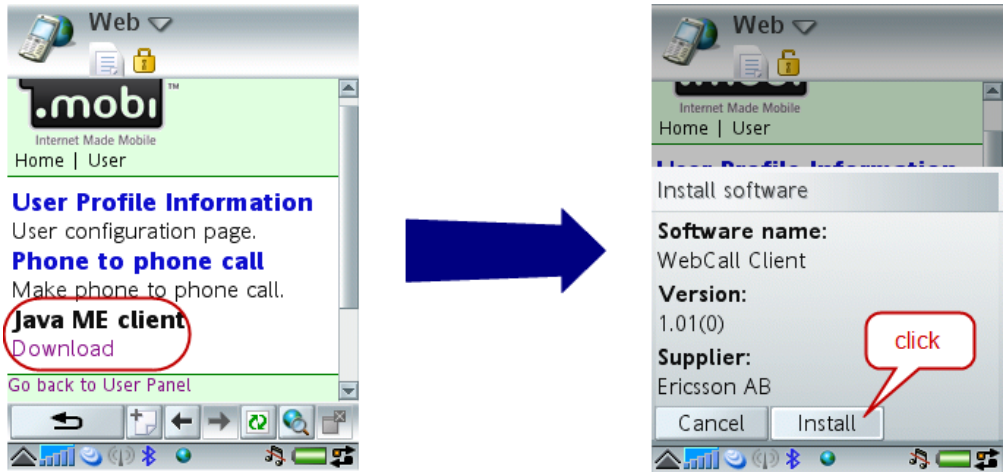

Figure 9.13. Download and install Java ME Client

9.9 Security of Java ME Client

All of the operation with web service interface need a user name and hashed password. Before download the Java ME Client from web application, the user have to login the web site. The user name and password is stored in JAD file as properties. The JAD file is automatically generated by the web application. The functionality of automatically generate JAD file is introduced in section **TODO: section number**. The JAD file will be saved in the memory of mobile phone, so users will not need to repeatedly input password when use the Java ME Client.

Chapter 10

Conclusion

Chapter 11

Future Work

11.1 RESTful web service interface

11.2 Support for more mobile devices

11.3 Gadgets

11.4 Call history

Bibliography

[1] Mary Campione and Kathy Walrath. The java programming language. http://java.sun.com/docs/books/tutorial/getStarted/intro/definition.html, February 2008. [cited at p. 4]

[2] Yuening Chen. *Web Call SDK*. Uppsala Universitet, March 2003. [cited at p. 7, 16]

[3] Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana. Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/wsdl, March 2001. [cited at p. 6]

[4] Jon Ellis and Mark Young. JSR-00172 J2ME Web Services Specification. http://jcp.org/en/jsr/detail?id=172, October 2003. [cited at p. 35]

[5] Elena Fersman and Peter Yeung. Using REST and Web Services to Mash Up Communications Capabilities. In *JavaOne 2009*, June 2009. [cited at p. 7]

[6] Soma Ghosh. J2ME record management store. http://www.ibm.com/developerworks/library/wi-rms/, May 2002. [cited at p. 4]

[7] Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, Henrik Frystyk Nielsen, Anish Karmarkar, and Yves Lafon. Soap Version 1.2. http://www.w3.org/TR/soap12, April 2007. [cited at p. 5]

[8] M. Handley, V. Jacobson, and C. Perkins. SDP: Session Description Protocol , RFC 4566. http://www.ietf.org/rfc/rfc4566.txt, July 2006. [cited at p. 5]

[9] Hugo Hass and Allen Brown. Web services glossary. http://www.w3.org/TR/ws-gloss/, February 2004. [cited at p. 5]

[10] Pär Johansson and Peter Yeung. *Mobile Front Controller Developer's guide for software version 3.1*. Ericsson Developer Connection, Ericsson, October 2008. [cited at p. 30, 58]

[11] Paul D. Kretkowski. State of the voip market 2008. http://www.voip-news.com/feature/state-voip-market-2008-031008/, March 2008. [cited at p. 11]

[12] Shanbo Li. Scallope.
http://scallope.googlecode.com/, February 2009. [cited at p. 20]

[13] Richard Marejka. Learning Path: MIDlet Life Cycle.
http://developers.sun.com/mobility/learn/midp/lifecycle/, February 2005. [cited at p. 4]

[14] Sun Microsystems. Java EE Technology.
http://java.sun.com/javaee/technologies/, June 2009. [cited at p. 4]

[15] Sun Microsystems. Java ME Technology.
http://java.sun.com/javame/technology/, June 2009. [cited at p. 4]

[16] Tim O'Reilly. What is web 2.0.
http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html, September 2005. [cited at p. 10]

[17] Java Community Process. JSR 75: PDA Optional Packages for the J2METM Platform.
http://jcp.org/en/jsr/detail?id=75, June 2004. [cited at p. 35, 38]

[18] Benny Ritzén. JavaOne: Come to Ericsson's BOFs and technical session and be inspired .
http://www.ericsson.com/developer/sub/articles/other_articles/090514_javaone_bof, May 2009. [cited at p. 7]

[19] J. Rosenberg, J. Peterson, H. Schulzrinne, and G. Camarillo. Best Current Practices for Third Party Call Control (3pcc) in the Session Initiation Protocol (sip), RFC 3725.
http://www.ietf.org/rfc/rfc3725.txt, April 2004. [cited at p. 12]

[20] J. Rosenberg., H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol, RFC 3261.
http://www.ietf.org/rfc/rfc3261.txt, June 2002. [cited at p. 5, 9, 18]

[21] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications, RFC 3550.
http://www.ietf.org/rfc/rfc3550.txt, July 2003. [cited at p. 5]

[22] R. Sparks. The Session Initiation Protocol (SIP) Refer Method.
http://www.ietf.org/rfc/rfc3515.txt, April 2003. [cited at p. 17]

[23] Wikipedia. Transport layer security.
http://en.wikipedia.org/wiki/Secure_Sockets_Layer, June 2009. [cited at p. 5]

[24] Wikipedia. Web 2.0.
http://en.wikipedia.org/wiki/Web_2.0, June 2009. [cited at p. 10]

[25] Peter Yeung and Pär Johansson. Mobile Front Controller.
http://www.ericsson.com, March 2009. [cited at p. 30]

Appendices

Appendix A

Appendix title

... some text ...

List of Symbols and Abbreviations

Abbreviation	Description	Definition
3GPP	3rd Generation Partnership Project	page 3
3pcc	Third Party Call Control	page 12
CS	Circuit Switched	page 3
GSM	Global System for Mobile communications	page ??
IMS	IP Multimedia Subsystem	page 3
IP	Internet Protocol	page ??
ISDN	Integrated Services Digital Network	page ??
ISUP	ISDN User Part	page ??
JDK	Java Development Kit	page ??
JRE	Java Runtime Environment	page ??
JVM	Java Virtual Machine	page 4
PSTN	Public Switched Telephone Network	page 6
QOS	Quality Of Service	page ??
RMS	Record Management System	page 4
SDP	Session Description Protocol	page 5
SIP	Session Initiation Protocol	page 5
UMTS	Universal Mobile Telecommunications System	page ??
VoIP	Voice over Internet Protocol	page 3

List of Figures

List of Tables