



Web Call Example Application

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

Contents	vii
1 Background	3
1.1 Introduction	3
1.2 Task	3
1.3 Terminology	4
1.4 About	7
2 Requirement	9
2.1 Programming Language	9
2.2 Simple	9
2.3 Stability	9
2.4 Reusability	9
2.5 Extendibility	10
2.6 Integration	10
3 Study	11
3.1 VoIP Market	11
3.1.1 VoIP Service Provider	11
3.1.2 VoIP Client	11
3.1.3 Solution Provider	12
3.2 Third Party Call Control	12
3.3 Why Web Call Example Application?	13
4 Solution	15
4.1 Relay Call	15
4.2 Problem of Relay Call	16
4.2.1 The Load on Controller	16
4.2.2 The Latency of Audio	16
4.2.3 Lack of Reliability	16

4.3	Third Party Call	16
4.3.1	Call Transfer	17
4.3.2	SDP Swap	18
4.3.3	Re-invite	19
4.3.4	Web Client	20
4.4	Conclusion	21
5	Application Overview	23
5.1	Use Cases	23
5.2	Architecture	25
5.2.1	SIP Call Component	25
5.2.2	Web Application	25
5.2.3	Web Service Interface	26
5.2.4	Java ME client	26
6	SIP Call Component	27
6.1	Five layers architecture	27
6.1.1	Protocol stack	27
6.1.2	Abstraction layer	27
6.1.3	Business logic layer	27
6.1.4	OOP use-case Based API Layer	27
6.1.5	JavaBean for synchronized communication	28
6.2	Five layers architecture in Web Call Example Application	28
6.3	Architecture of SIP Call Component	29
6.4	Class Hierarchy	30
7	Web Application	33
7.1	Overview	33
7.2	Architecture of Mobile Front Controller 3	34
7.3	Dual view	35
7.3.1	Desktop browser view	35
7.3.2	Mobile browser view	36
7.4	Site structure	36
7.5	User action	37
7.5.1	User Registration	37
7.5.2	User panel	37
7.5.3	VoIP service provider account	38
7.5.4	Phone-to-Phone call	38
7.6	Administrator action	40
7.7	Validation mechanism	41
7.7.1	Page level	41
7.7.2	Server level	42

7.8	Session control	42
7.9	Ajax in web application	42
7.10	Java ME helper	43
7.11	Database	43
7.11.1	Design of database	43
7.11.2	User database utility	43
7.12	Security	45
8	Web Service Interface	47
8.1	Introduce web service and metro	47
8.2	SOAP web service	47
8.3	Deploy of Web Service	49
9	Java ME Client	51
9.1	Architecture	51
9.2	Web Service Client Stub	52
9.3	Record Store Manager	53
9.4	PIM Contact Helper	53
9.5	Web Call Client	54
9.6	User Interface	56
9.6.1	VoIP Call Form	56
9.6.2	PIM Contacts	57
9.6.3	My Phone Number	57
9.6.4	Contacts	58
9.6.5	Synchronize	58
9.6.6	Phone Call	58
9.7	Two implementations of UI	60
9.7.1	VMD-based UI	60
9.7.2	Generic UI	60
9.8	Installation of Java ME Client	62
9.9	Security of Java ME Client	62
10	Conclusion	63
11	Future Work	65
11.1	RESTful web service interface	65
11.2	Support for more mobile devices	65
11.3	Gadgets	65
11.4	Call history	65
	Bibliography	67
A	Appendix title	73

List of Symbols and Abbreviations	75
List of Figures	76
List of Tables	78

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 Example Application(former *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]

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.[17]

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.[18]

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.[8]

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.[16]

*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.[28]

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.[25]

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.[11]

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.[26]

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. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.*”.[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.[10]

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 Example Application is an open source project at **Ericsson Developer Connection**[†] (EDC), **Ericsson**[‡]. 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 Example Application**. 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 Example Application** is used as the base library of Ericsson's demo of **Using REST and Web Services to Mash Up Communications Capabilities**[5][23] at **JavaOne**^{TM§} 2009.

[†]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.

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

[§]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 portable, **Java** is chosen as the programming language of Web Call Example Application. 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 Example Application should supply 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 [25], 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 Example Application.

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 Example Application should supply a web service API that can be used by other applications. This interface should contain most of the functions of Web Call Example Application. And can be easily used on Web 2.0* mashup[†].

*“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.[29] See also: [What Is Web 2.0](#)[21]

[†]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.^[14]

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^{TM*}, VoipStunt[†] and JAJAH[‡]. 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.[24]

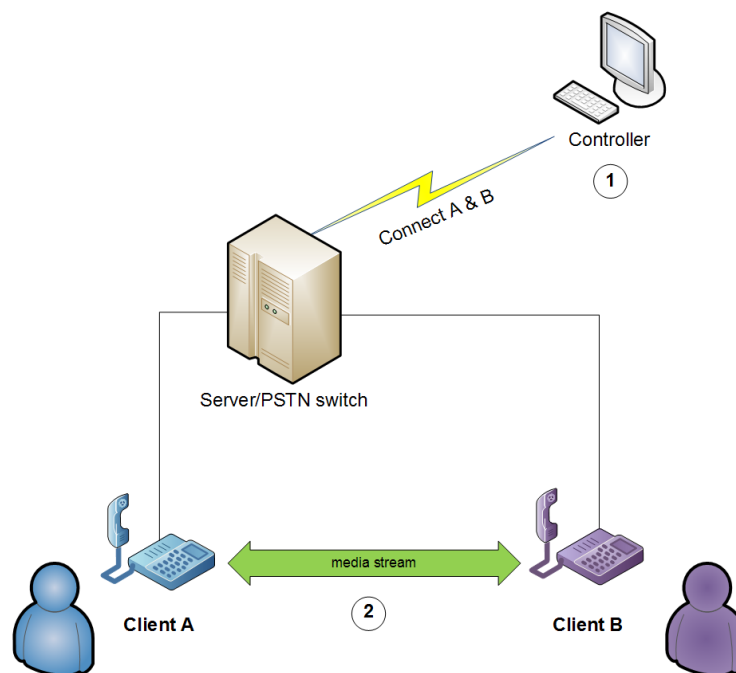


Figure 3.1. Third party call control

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

*<http://www.skype.com>

†<http://www.voipstunt.com>

‡<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 Example Application?

Based on IMS/SIP technology, Web Call Example Application 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 Example Application is neither 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 Example Application 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

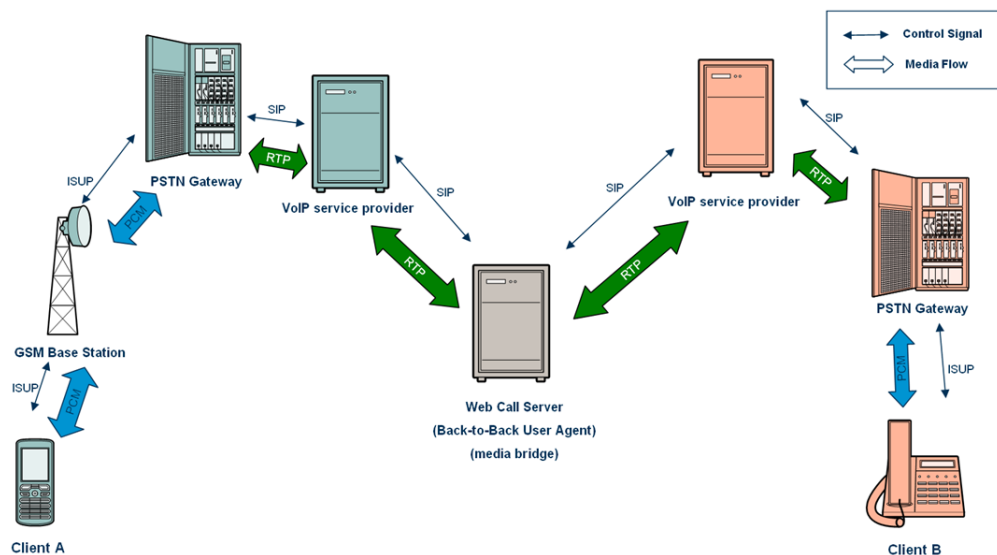


Figure 4.1. The signal and media flow of Relay Call

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 than 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.

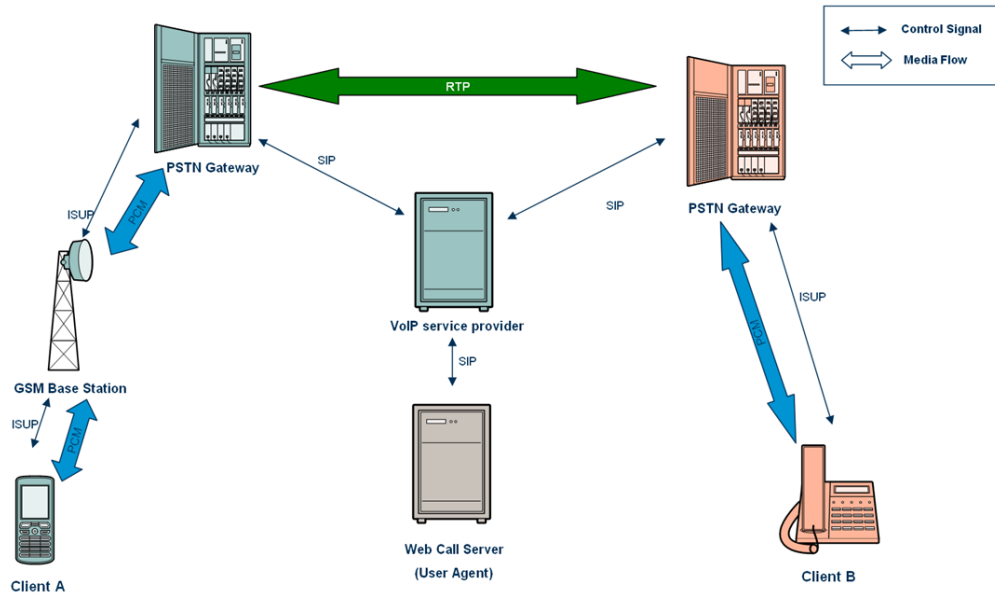


Figure 4.2. The signal and media flow of Third Party Call

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)[27]. “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”[27].

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.

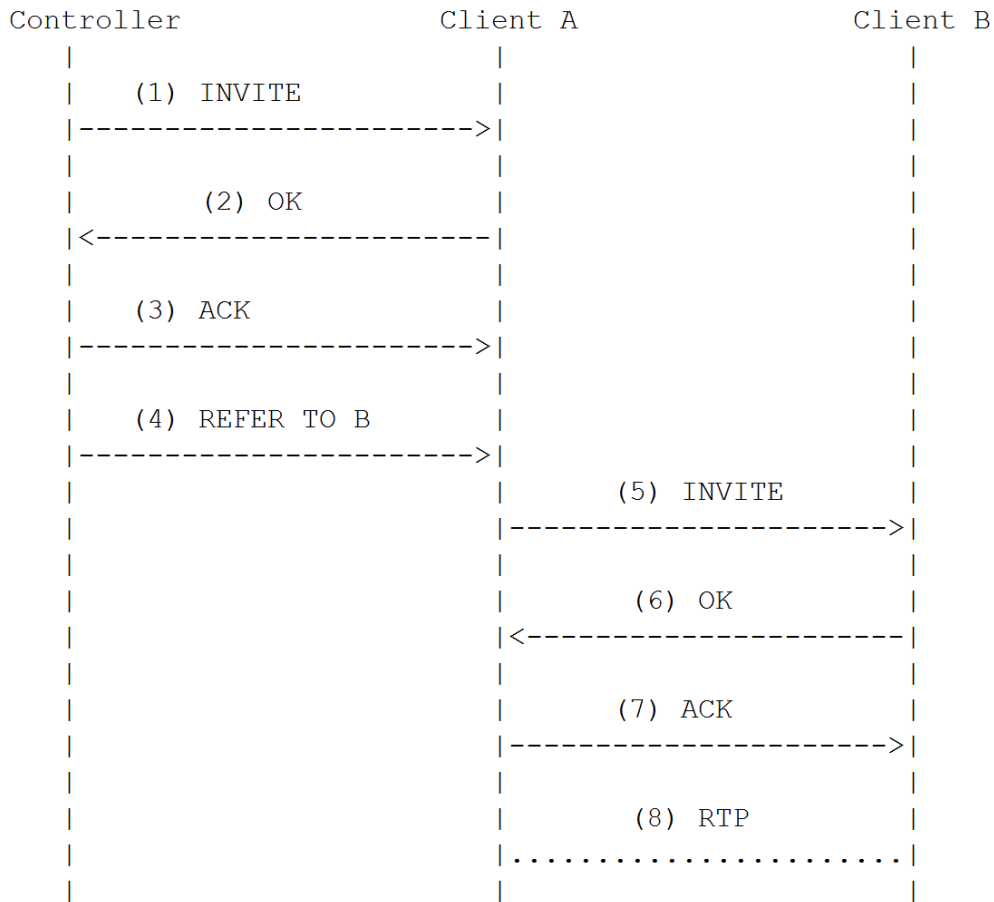


Figure 4.3. The signal and media flow of Call Transfer

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 Swap** 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 [25]. 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).

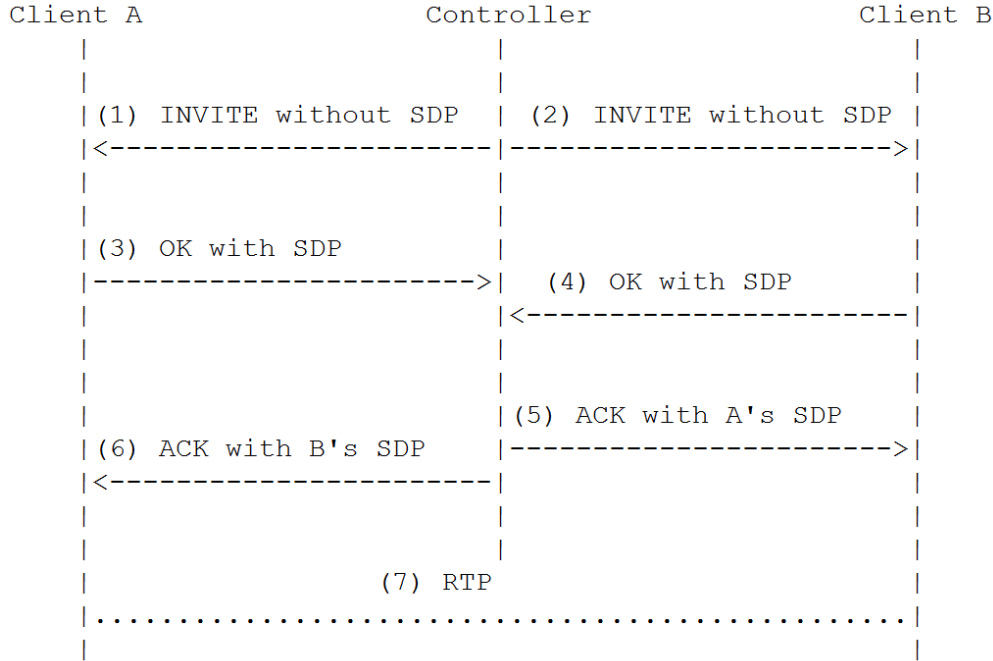


Figure 4.4. The signal and media flow of SDP Swap

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).

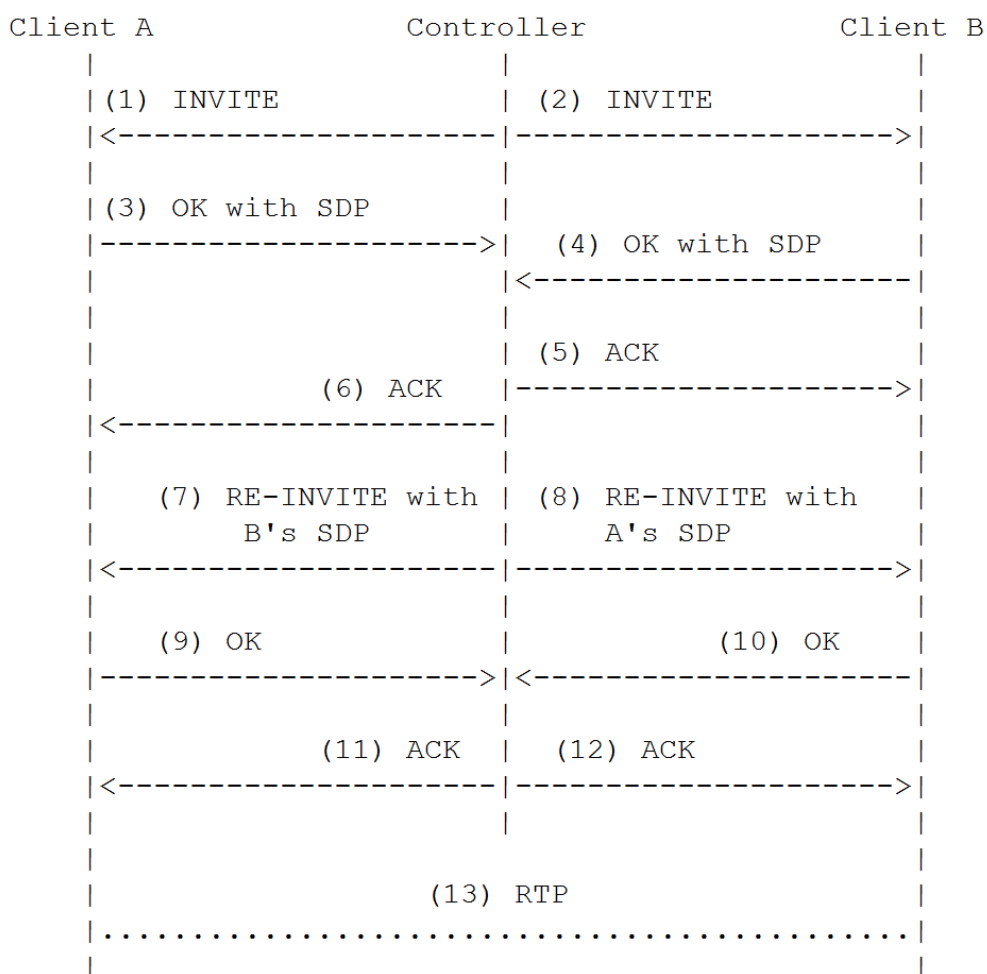


Figure 4.5. The signal and media flow of SDP Swap

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**[15]. 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.

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

Table 4.1. Comparison of Call Method

Chapter 5

Application Overview

5.1 Use Cases

Web Call Example Application is a application that resides at a web server and supplies functionality of VoIP phone calls.

As shown in Figure 5.1, there are three use cases. They are Desktop Browser, Mobile Browser and Java ME Client. This makes it possible for users to make VoIP calls anytime anywhere, to anyone. The use case of Desktop Browser is used when user have a computer connected to Internet. He can use his desktop browser, e.g. Firefox or Chrome, to access the web call site by inputing URL to address bar of browser. Users can use this web site to make VoIP phone calls as well as manage his account. The use case of Mobile Browser is just a web site for mobile browsers. The function is the same as Desktop Browser. The user can also manage his account from the Mobile Browser view. Compare with Mobile Browser, the use case of Java ME Client gives the user a even easier way to make VoIP phone calls. There is no need to sign in and input any information. As long as user pre-input everything on web site, the Java ME Client will retrieve account informations the server. All user need to do is just click the "call" button and enjoy the cheap VoIP phone calls. The Java ME Client can also load mobile phone's native contact book and call the people in the list. It can also synchronize the contact book of mobile phone to server. So user can use the contact book later when he sitting in front of a computer. A user could have multi-account for different VoIP service providers. So when the user want to make a international call, he could choose a cheap cost service provider according different rates among providers.

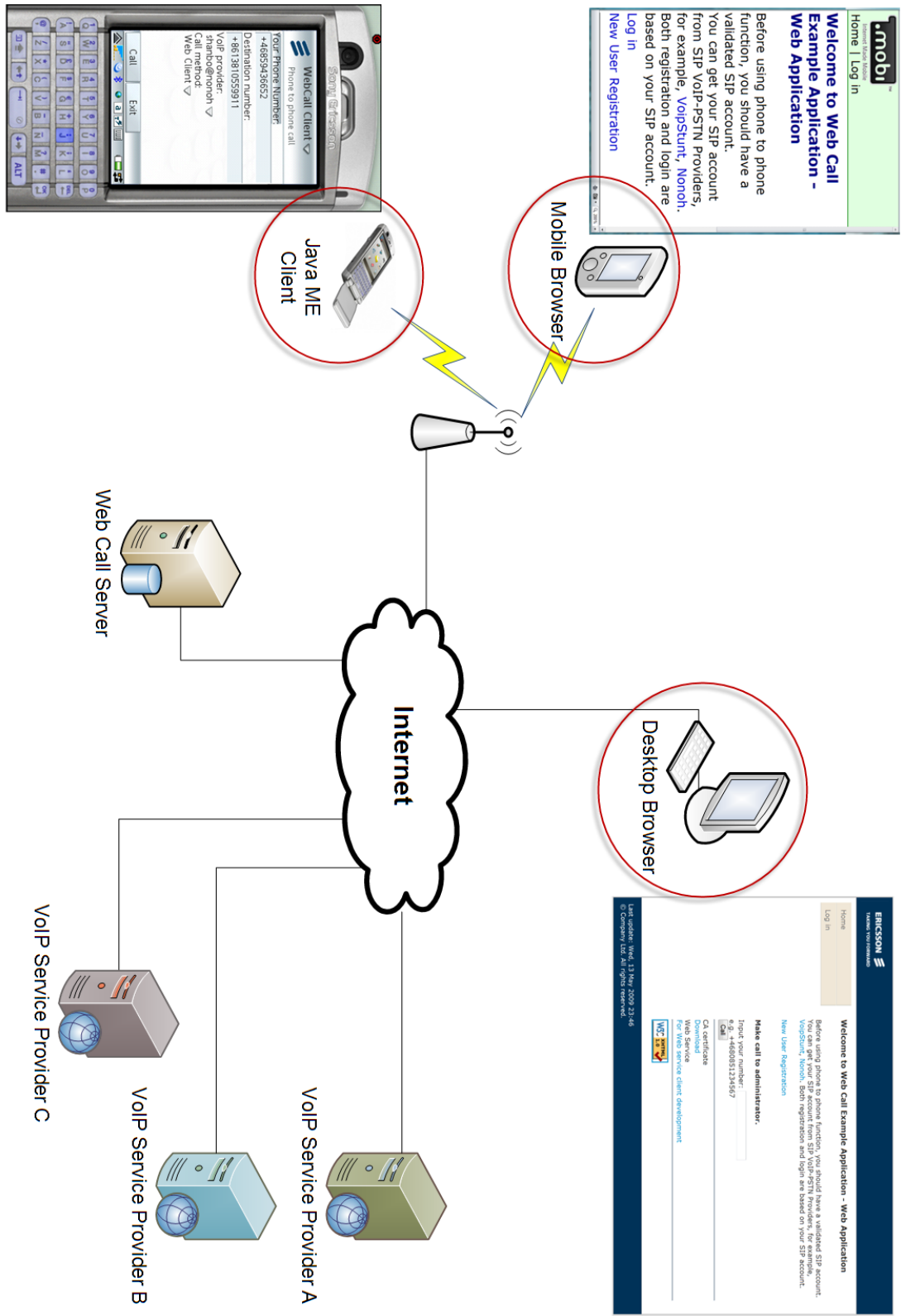


Figure 5.1. Use Cases of Web Call Example Application

5.2 Architecture

This chapter presents a brief solution description of Web Call Example Application. The design architecture is shown in Figure 5.2.

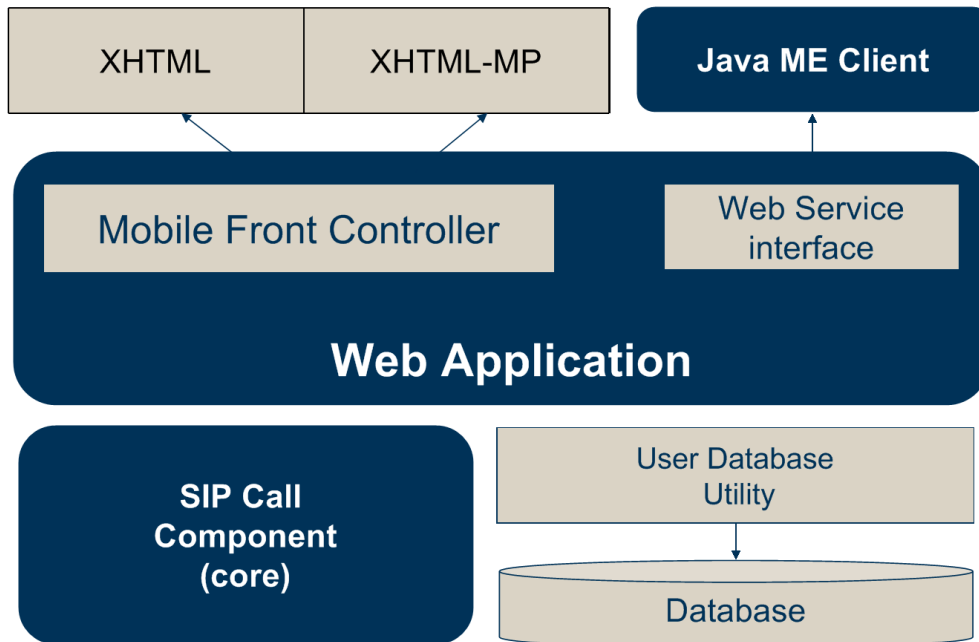


Figure 5.2. The Architecture of Web Call Example Application

Web Call Example Application 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 5.2.

5.2.1 SIP Call Component

The SIP Call Component is the core of Web Call Example Application. 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.

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. This five layers architecture is developed by *Peter Yeung* from Ericsson Developer Connection.

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 Business 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.

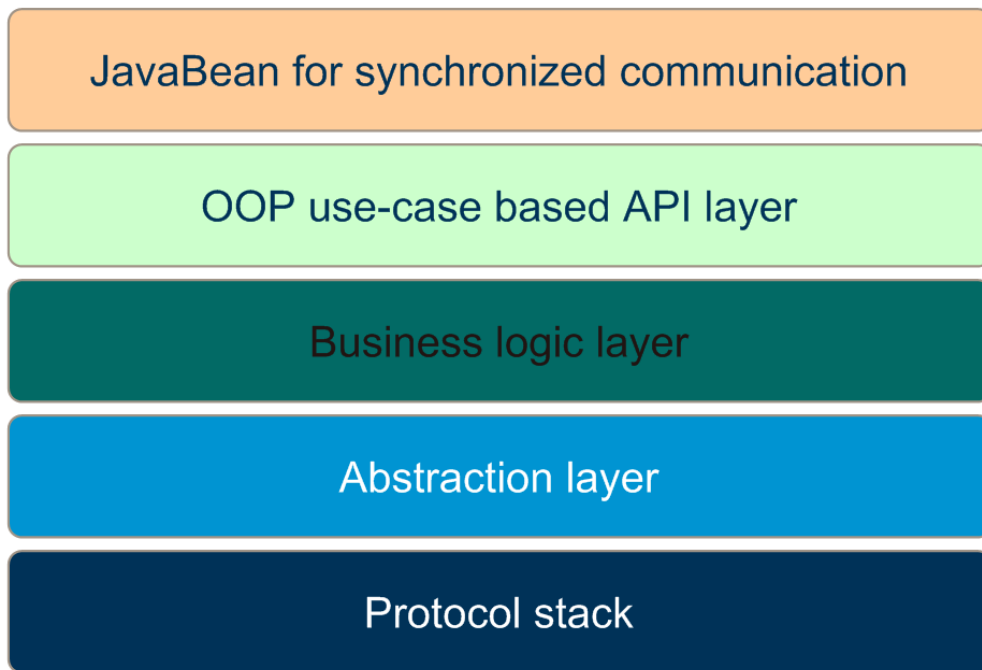


Figure 6.1. Five layers architecture

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 Five layers architecture in Web Call Example Application

Web Call Example Application follows the concept of five layers architecture.

In SIP Call Component, there is abstraction layer and protocol stacks. As is shown in Figure 6.3, Among the four implementations of Third Party Call Controller, the left three implementations are based on MjSIP and Web Client is based on Apache Common HttpClient. A advantage of separate logic and Protocol stack is that users can choose to use different stacks according to their requirements. Another advantage is that developers can focus on the business logic part and no need to touch lower level stacks. In SIP Call Component, the abstraction layer

defined a general interface for normal phone call behaviors, such as `start()` and `terminate()`. The interface of abstraction layer in SIP Call Component is shown in Figure 6.2. As the business logic of SIP Call Component, Call Controller only needs to take care the call flow, e.g. set properties, register, make phone call and terminate call. And it will not handle any details about SIP message or media flow from the API level. The public API that opens to developers in SIP Call Component is the “OOP use-case Based API Layer”.

In Web Call Example Application, SIP Call Component is used as the base library of Web Application. However, It is designed to use as a stand alone SIP call high level API. It is independent with other parts of the application. Any developers can take it and integrate it to their own applications.

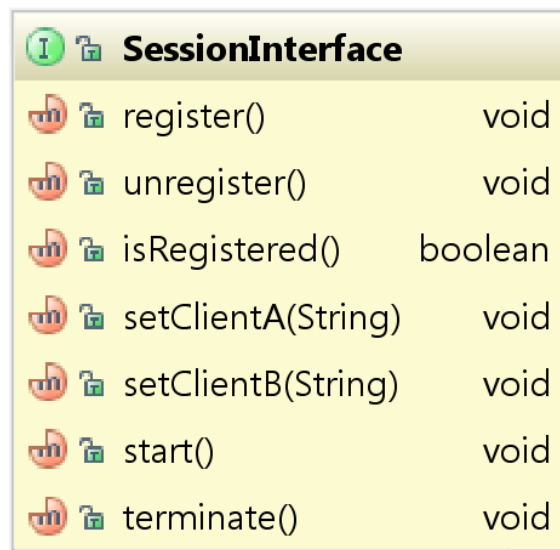


Figure 6.2. Class diagram of `SessionInterface`

6.3 Architecture of SIP Call Component

The SIP Call Component is the core of Web Call Example Application. The design architecture is shown in Figure 6.3.

It can be seen from Figure 6.3, There is a interface `CallController` on the top of all kinds of call controllers. There are two subtype of Call Controller, one is Relay Call Controller and another is Third Party Call Controller. The Relay Call Controller is the legacy code of Web Call SDK which developed by Yuening Chen. In the old project, the Relay Call Controller is the core of all phone calls[2]. However, in this version, four other implements are added to the SIP Call Component. They are Call Transfer, SDP Swap, Re-invite and Web Client. From a out side view, all of the four methods are doing the same thing. They just

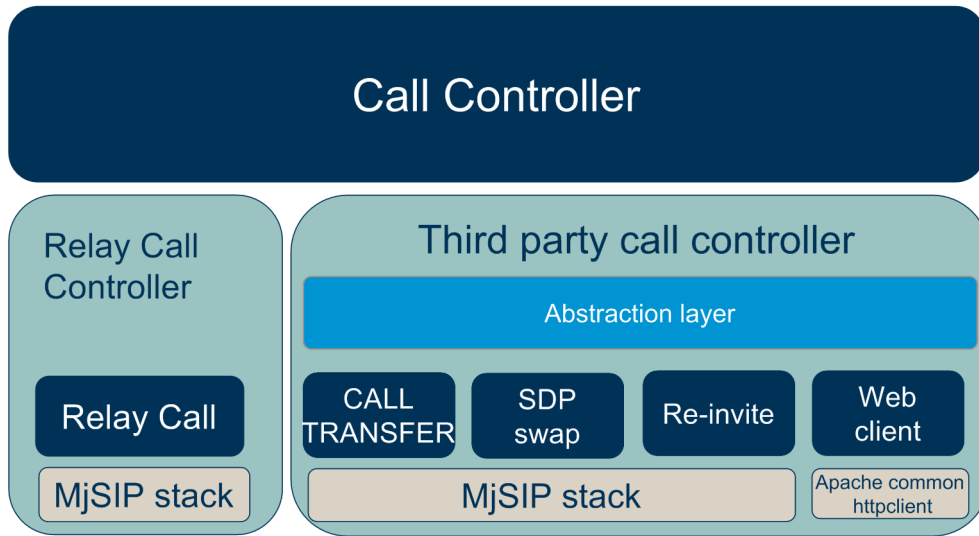


Figure 6.3. The Architecture of SIP Call Component

try to establish a phone call between caller and callee. Chapter 4 describes the different among the four methods in detail. Chapter will focus on the architecture and design part. All of four third party call methods are implemented the same interface. This means the user can easily switch call method from one to another. And other call method, if there is any, as long as it implements the interface at abstraction layer. It can be used in the SIP Call Component.

6.4 Class Hierarchy

The hierarchy of call controller is shown in Figure 6.4. All of call controller is implemented the first interface `CallController`. This kind of design enables easily switch of call controllers. `SessionInterface` is a interface for third party call. `WebClientImpl`, `ReinviteImpl`, `SdpSwapImpl` and `MjSipCallTransferImpl` are implementations for `SessionInterface`.

There are different call controller factories to create call controllers. The call controller factory follows the design pattern of `Abstract Factory`. An `Abstract Factory` is a class that exists to create instances of another class[6]. The design of SIP Call Component considered that a call controller can not only establish audio call, but also video call and message channels. This can be seen from the Figure 6.5. The very general difference between Relay Call and Third Party Call is the way to handle media flow. But both of them should have the competence to establish all kind of sessions. In current version of Web Call Example Application, only audio call is implemented.

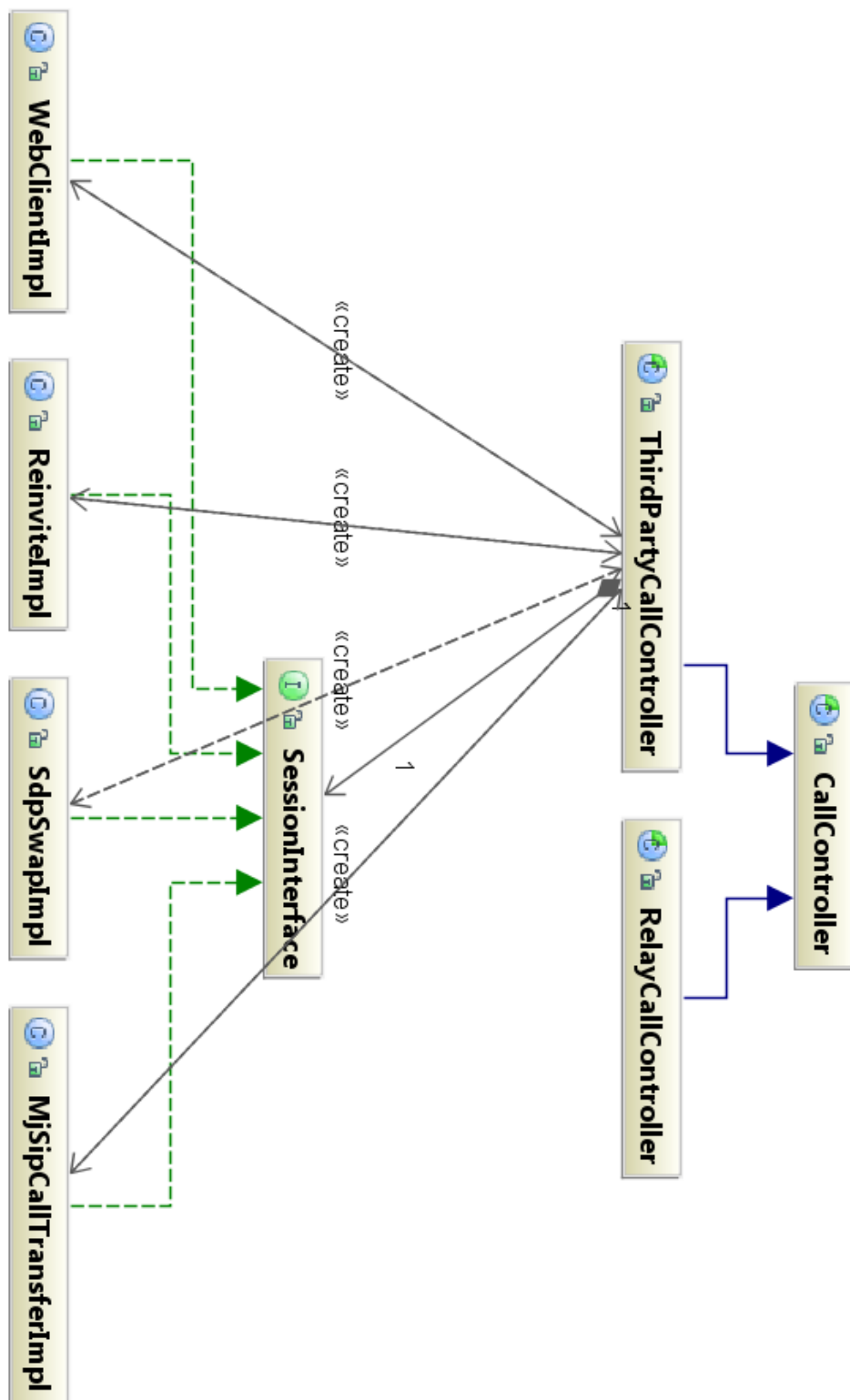


Figure 6.4. Call Controller class hierarchy diagram

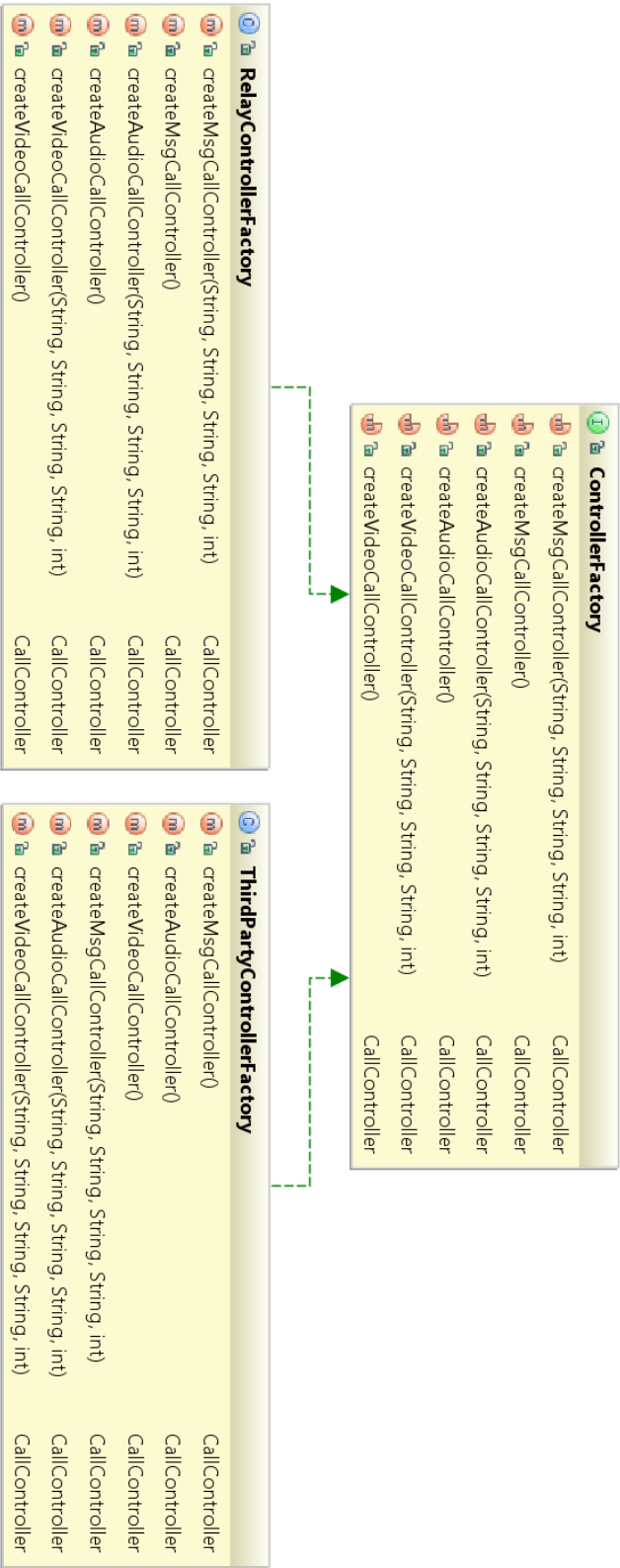


Figure 6.5. Controller Factory class hierarchy diagram

Chapter 7

Web Application

7.1 Overview

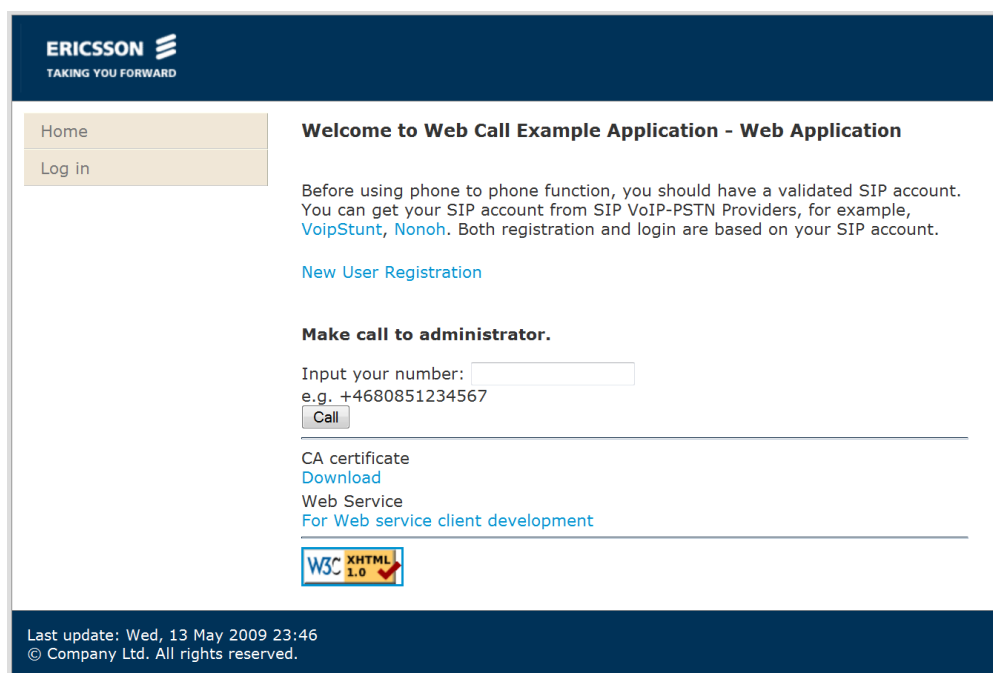


Figure 7.1. Welcome page of desktop browser view of web application

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 5.2. 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 servlet 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.”[30] It was developed by Peter Yeung and Pär Johansson from [Ericsson Developer Connection](#), Ericsson[30]. The mobile front controller uses a servlet to handle http request, and redirect request to different kind of view. All views share a same logic. An overview of Mobile Front Controller used by a web application on a Java EE web container is shown in Figure 7.2.

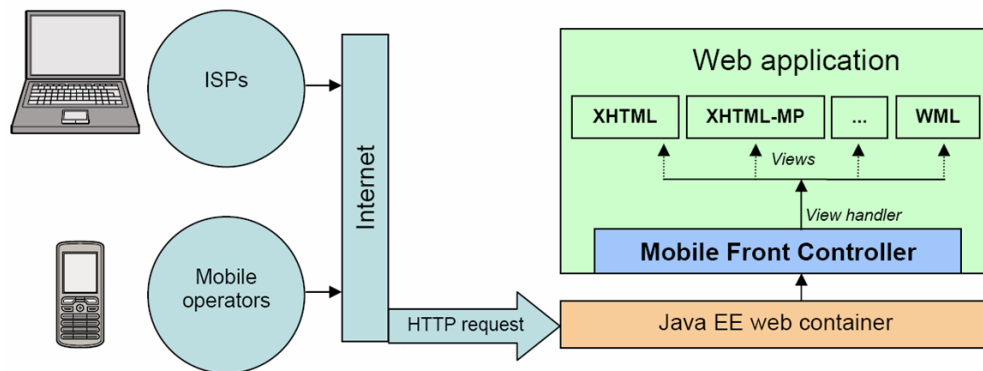


Figure 7.2. An overview of Mobile Front Controller used by a web application on a Java EE web container. (Figure taken from *Mobile Front Controller Developer’s guide for software version 3.1*[13])

“MFC is used on top of a Java EE web container, and does not require any other framework, such as JSF, Struts, etc. MFC does the following:

- Detects and selects appropriate views based on HTTP request headers. A view is a subdirectory that, for example, corresponds to a markup language such as XHTML, XHTML Mobile Profile (MP) and WML (see Figure 7.3). The way MFC detects and selects views is customizable using view handlers.
- Shares UI logic between different views, for example, web and mobile browsing. This is done using action commands, which are classes that contain an execute method that is executed when a URL with, for example, the

*URL pattern *.do is called (for example `http://localhost:8080/mfc-basic-demo/xhtml/Print.do`).”[13]*

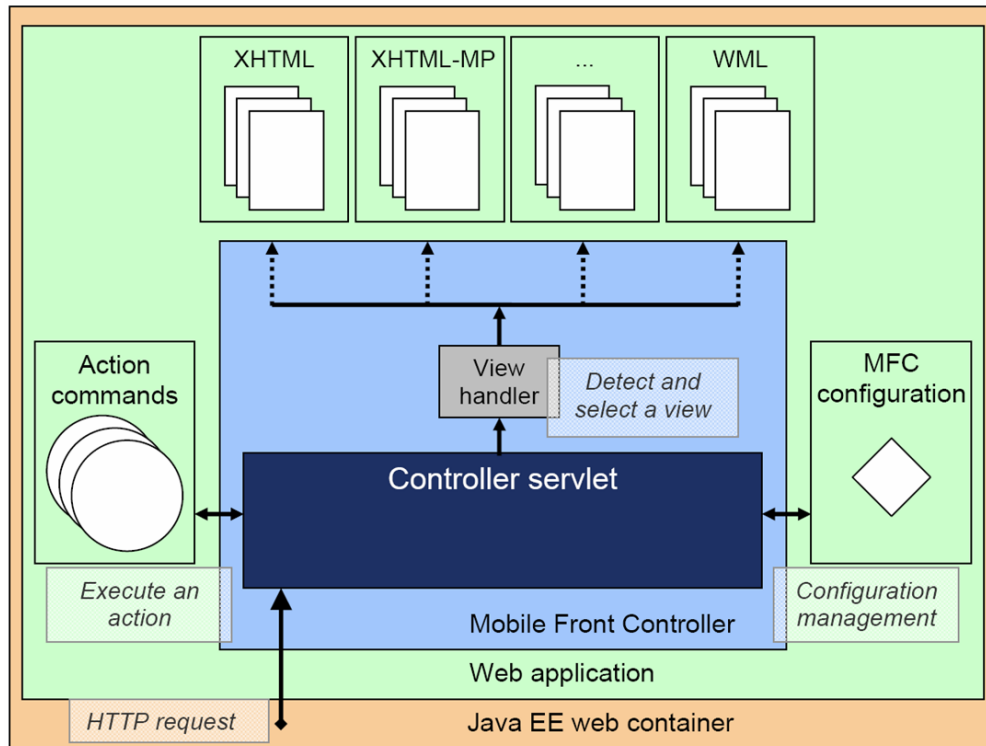


Figure 7.3. The architecture of Mobile Front Controller (Figure taken from *Mobile Front Controller Developer's guide for software version 3.1*[13])

The version of Mobile Front Controller used in Web Call Example Application is the latest version, v3.1.

7.3 Dual view

As described in the section 7.2, Mobile Front Controller supplies multi-view function. Web Call Example Application is developed with two view, one is desktop browser view and the other is mobile browser view. The will be illustrated separately in subsection 7.3.1 and 7.3.2.

7.3.1 Desktop browser view

The Mobile Front Controller supplies a function of select view. If the user access web application via a desktop browser, the desktop browser view (XHTML view) will return to user. The desktop browser view is a normal web site that follows the standard of The XHTML 1.0. Extensible HyperText Markup Language (XHTML) is a markup language for web pages which developed by W3C

HTML Working Group. It is widely used as a standard language of web page on Internet. A example page (welcome page) of desktop browser view is shown in Figure 7.1.

7.3.2 Mobile browser view

If the user access web application via a mobile browser, the mobile browser view (XHTML MP view) will return to user. The mobile browser view is a normal web site that follows the standard of XHTML MP 1.1. XHTML Mobile Profile (XHTML MP) is a computer language standard designed specifically for mobile phones. It is developed by A example page (welcome page) of mobile browser view is shown in Figure 7.4.

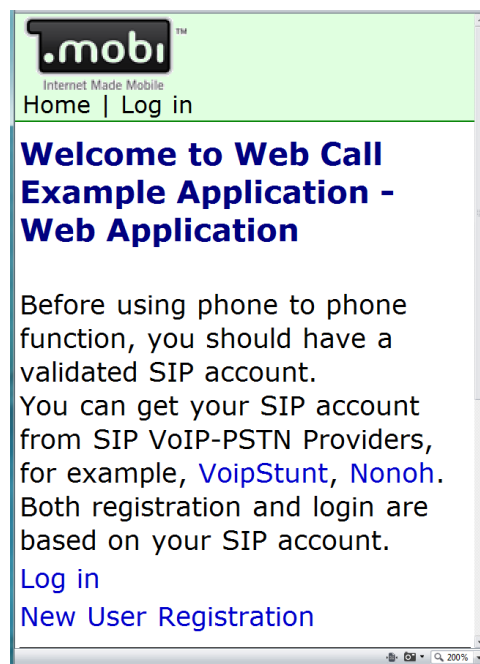


Figure 7.4. Welcome page of mobile browser view

7.4 Site structure

The web set structure contains three levels. **Base** → **Protected** → **admin**. Both desktop browser view and mobile browser view use the exactly same structure.

Under base directory, there is welcome page, user register page and login page. The pages on this level are public opened. There are no restrictions on requesting these pages.

All pages in protected directory are authenticated resources. To visit these pages, the user has to login first. Both user and administrator can request this

kind of resources. Under user panel, user can freely edit information, e.g. user phone number, contact book, change password, VoIP provider account and recent calls. Users can make VoIP calls by clicking the link of Phone to Phone Call under user panel. The pages under admin directory are used for managing user information. Only user who has a role of administrator can access pages under admin directory. Administrator can modify user information or even delete users. For more details about security, please refer section 7.12.

7.5 User action

7.5.1 User Registration

To use the Web Call Example Application, a user must register him on the web site. It is quite easy and simple to register and there are only three steps:

1. Navigate to the web site url and open the welcome page. There are user registration functions on both desktop and mobile views. So users can register either from a computer or a mobile phone.
2. At the welcome page. Click the link to New User Registration.
3. Fill the form, include user name (in the format of email address), password and confirm password.

7.5.2 User panel

The user profile information menu is shown in Figure 7.5. User can edit his phone number, contact book, password, VoIP provider accounts and recent calls log.



Figure 7.5. User profile information page of desktop browser view

7.5.3 VoIP service provider account

To make a VoIP call, the user must have a VoIP Provider Account first. Figure 7.6 shows how to register a VoIP provider account. To get the view of adding VoIP provider account, user should go to user panel → User Profile Information → VoIP Provider Account → add new VoIP provider account. Follow the example in Figure 7.6 and input the VoIP account information. The outbound proxy should in a format of SIP_PROVIDER_URL:PORT_NUMBER. If no port number specified, the port 5060 will be used as defined in RFC3261[25].

ERICSSON
TAKING YOU FORWARD

Welcome, shanboli

Home
Log out
User panel

Add a new VoIP provider account

account name *: shanbo@nonoh.net e.g. tom@sip.sipprovider.com

SIP username *: shanbo e.g. tom

SIP password *: e.g. k8g3A8z

confirm password *: input the same password again.

realm *: sip.nonoh.net e.g. sip.sipprovider.com

outbound proxy *: sip.nonoh.net5060 e.g. sip.sipprovider.com

Local port *: 5099 The port which the sip agent will use e.g. 5060

Port A: 11240 This is only used for Relay Call e.g. 11240

Port B: 11241 This is only used for Relay Call e.g. 11241

All fields marked with * are mandatory.

Add Cancel

Last update: Sun, 15 February 2009 17:09
© Company Ltd. All rights reserved.

Figure 7.6. Register VoIP service provider account

Web Call Example Application supports multi-account. This means as a user, you can have more than one account and choose a cheaper one when you make VoIP calls.

7.5.4 Phone-to-Phone call

The integrated web application offers phone-to-phone function for registered users.

Figure 7.7 is the user panel of desktop browser view. Users can either go to user profile Information page or the phone-to-phone dialing page. To make VoIP calls, the user just need click the link of Phone to phone call.

In the phone-to-phone call page, which is shown in Figure 7.8, users can input their own number, or use the default numbers, which is set in “my phone number” of user profiles information. And input the destination phone number, or select from recent calls or the contact list. The client number could either be SIP address or a complete phone number. If users have more than one VoIP provider account, they can use the drop down list of VoIP provider account and choose



Figure 7.7. User panel desktop browser view

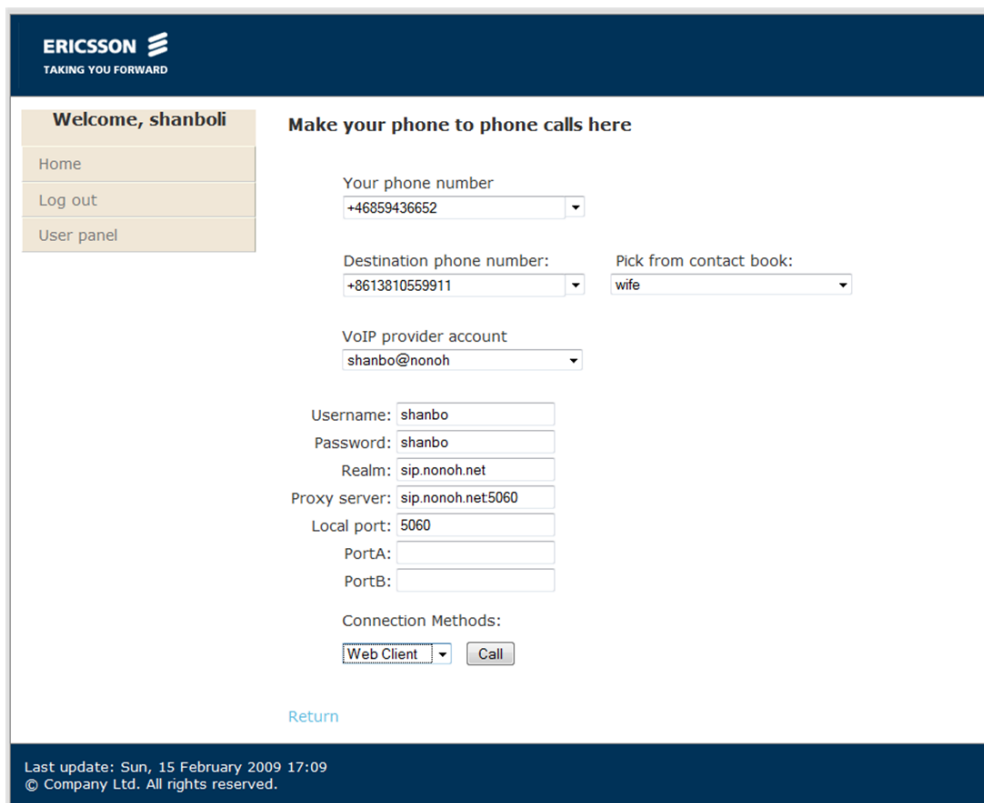


Figure 7.8. Phone to phone call


an account as their wish. Different VoIP service provider may supply different rate for different destination. So the multi-account function can help user save money by choosing a cheapest provider. This page uses the technology of Ajax to dynamically show the information of VoIP account. More Ajax technologies will be introduced in section 7.9.

There is a drop down list of call methods. Five different implementations of connection methods will be illustrated at that list. They are Relay Call, Call transfer, Re-invite, SDP swap and Web client. The last four methods are so called third party call control (3PCC)[24]. The differences of connection methods will be discussed in chapter 4. It is recommend that by default user uses the method of Web client.

After user clicked “Call” button. The page will be directed to call state page. On call state page, users can see the phone numbers of two sides, call state, or even cancel the call.

7.6 Administrator action

The administrator actions include user profile management as well as all of user relevant information. Figure 7.9 is the user list page. The administrator can use the button of edit to change the user information as well as user roles.

ERICSSON 
TAKING YOU FORWARD

Welcome, admin

Home

Log out

Admin panel

User panel

User List

username	password	role		
admin	d033e22ae348aeb5660fc2140aec35850c4da997	[USER, ADMIN]	Edit	Delete
a	6dcd4ce23d88e2ee9568ba546c007c63d9131c1b	[USER]	Edit	Delete
b	e9d71f5ee7c92d6dc9e92ffdad17b8bd49418f98	[USER]	Edit	Delete
shanboli	ab0e154dccb95f866d5e21dac51afc25d0a49c75	[USER]	Edit	Delete

[Return](#)

Last update: Mon, 23 February 2009 15:50
© Company Ltd. All rights reserved.

Figure 7.9. User list of administrator panel

Figure 7.10 shows a administrator view of user info. The administrator can update user’s basic info, E.g. password and roles. Administrators can change SIP relevant information, such as user’s phone number, user’s contact list and user’s VoIP account info by clicking “show” button in the SIP relevant info section.

The administrator can not delete the last administrator, otherwise no one can manage the application.

ERICSSON
TAKING YOU FORWARD

Welcome, admin

Home

Log out

Admin panel

User panel

User Info

Basic info

Field	Value	New Value
Username:	shanboli	
Password:	ab0e154dcc95f866d5e21dac51afc25d0a49c75	
Role(s):	[USER]	<input checked="" type="checkbox"/> USER <input type="checkbox"/> ADMIN

Update

SIP relevant info

Show

[Return](#)

Last update: Mon, 23 February 2009 15:50
© Company Ltd. All rights reserved.

Figure 7.10. Administrator view of user information

7.7 Validation mechanism

The validation of web application is implemented both at presentation tier and logic tier. On presentation tier, all form are validated by javascript.

7.7.1 Page level

The validation on presentation layer can save the time of communication between browser and server. Web Call Example Application uses JavaScript to validate user input. All pages share the same generic validation methods. An example code of “checking if a field is empty” is shown in Listing 7.1.

Listing 7.1. JavaScript validation code

```
function has_value(field , alert_message) {
    if (field.value == null || field.value == "") {
        alert(alert_message);
        return false;
    }
    return true;
}
```

7.7.2 Server level

However, to prevent user disables `javascript`, there is also a validation mechanism on server side, that is called server level validation. To make the server response faster, only important or sensitive actions use a server level validation. E.g. The action of changing password has a server level validation of checking if password matched. The action of cancel call is another sensitive function. Beside the session id, server will also check if this session belongs to the user who want to cancel it. This will prevent from arbitrary interrupt calls.

7.8 Session control

Web Call Example Application supports multi thread actions, that is many users can make phone calls concurrently. An `enum` class `CallSessions` works as a singleton* and manages call sessions. A class diagram of `CallSessions` is shown in Figure 7.11. It can be seen from picture, the four methods `add`, `getCallInfo`, `remove`, `contains` supply a very convenient way to manipulate sessions. A instance of `CallInfo` contains all of information of a phone call, it can be treated as a information pack.

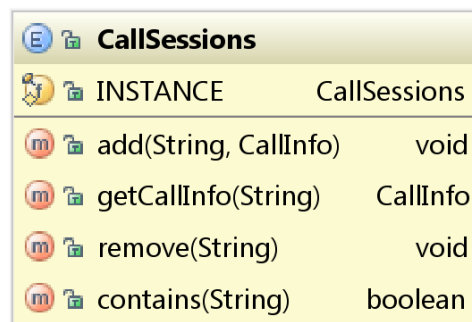


Figure 7.11. Class diagram of `CallSessions`

7.9 Ajax in web application

Ajax is short for Asynchronous JavaScript and XML. It is a set of Internet techniques used on the client to generate interactive web applications or rich Internet applications. For more detail about Ajax, please refer to *Ajax : A New Approach to Web Applications* by Jesse James Garrett[7].

There are two places use Ajax in Web Call Example Application. One is at phone to phone call page, which is shown in Figure 7.8. When user chooses a different VoIP service provider account. An Ajax request will be used to fetch

*Singleton is a design pattern that used to restrict instantiation of a class to one object.

account information from server. The fields will be updated automatically after browser get new account info. And there is no refresh of the page.

Another place that uses Ajax is the call state page. The call state page shows a real time state of a call session. There is a **JavaScript** function named **updateState** sends ajaxGet request to server to check the session state in every minute. An action named **LoadCallState** at server side will communicate with **CallSessions** which described in section 7.8, and response browser with the latest state of that session.

7.10 Java ME helper

Java ME helper is a servlet which responses a content of JAD file to mobile devices. It is not simply load a JAD file and send content to client. Instead, it automatically generates the content of JAD file on demand. Normally, a JAD file contains the description of JAR and MIDlets. Beside the standard properties, the Java ME helper will also collect the user name and hashed password and write it into JAD. The username and hashed password will be used in communication of Java ME Client which will be introduced in chapter 9.

7.11 Database

The database of Web Call Example Application stores all user informations. When users login to the web page they will always be asked to input their user name and password. The inputted user name and password will be compared with the ones in database.

7.11.1 Design of database

A EER(Extended Entity-Relationship) diagram of database is shown in Figure 7.12. There are seven tables in the database. They are **user**, **role**, **user_role**, **contact**, **user_uri**, **sip_account** and **recent_call**. Table **user** stores user name and hashed password. Table **role** stores typical roles in the application, that is USER and ADMIN. Table **user_role** is a bridge table that presences a many to many relationship of table **user** and **role**. Table **contact** stores the contacts informations. Table **user_uri** stores primary phone numbers of user. Table **sip_account** stores VoIP service provider account that is used to register to a proxy and make phone calls. Table **recent_call** stores recent call history.

7.11.2 User database utility

User database utility is originally a part of Web Call Example Application. It was separated from Web Call Example Application and distributed as an example

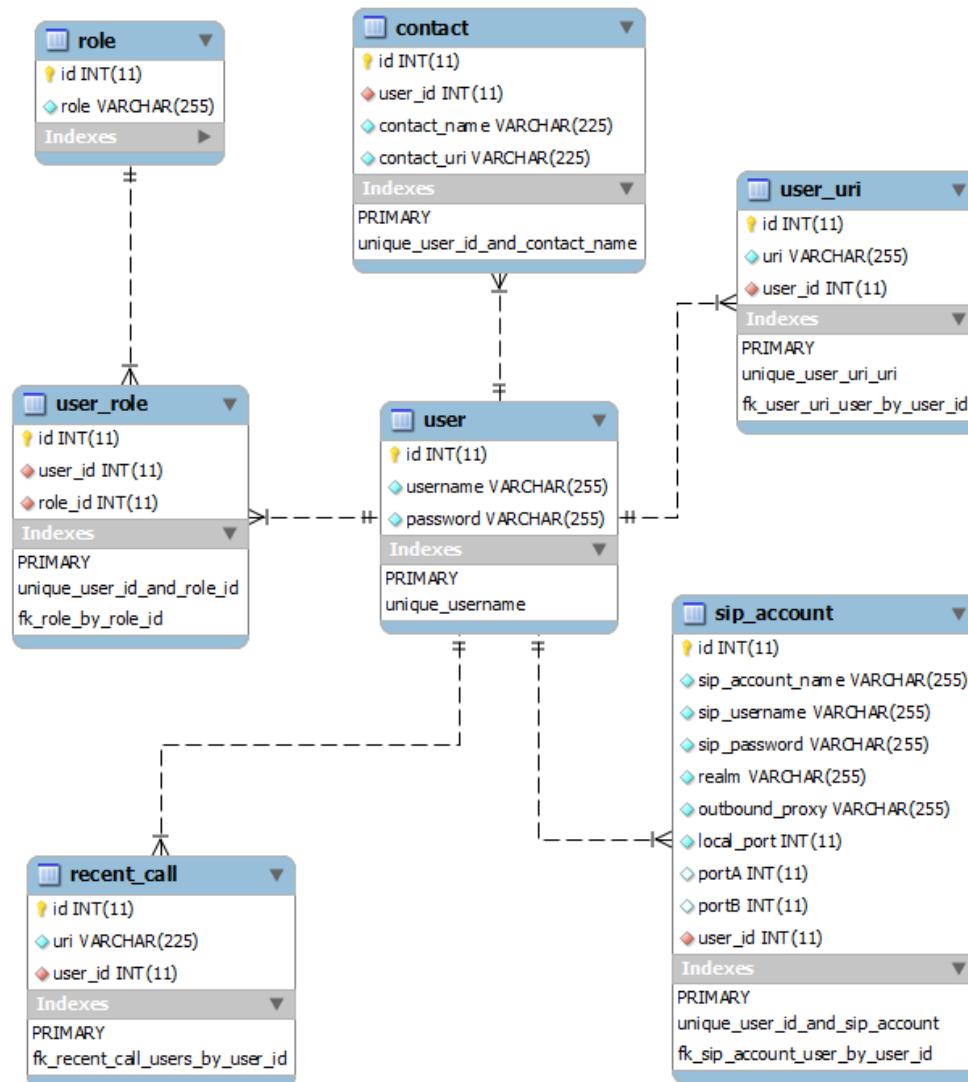


Figure 7.12. EER diagram of database (Diagram generated by MySQL Workbench)

of Mobile Front Controller. The User database utility supplies a light weight java API for manipulation database. The user database utility uses a Strategy Pattern[†] to make user easy to interchange database connection from data source to direct connection.

As long as user creates a database, he can use Web Call Example Application. When the first time deploys, it will check if the database has the tables which it needs. If it is an empty database, Web Call Example Application will automatically generates essential tables. And the first user of this application will be automatically registered as administrator, which is shown in Figure 7.13

Figure 7.13. First user will be administrator

7.12 Security

There are two parts of resources are in the security constraint domain. One is the user resource and another is administrator resources. They are both in the CONFIDENTIAL level. *“A user data constraint (<user-data-constraint> in the deployment descriptor) requires that all constrained URL patterns and HTTP methods specified in the security constraint are received over a protected transport layer connection such as HTTPS (HTTP over SSL).”* from Java EE 5 Tutorial[12].

Web Call Example Application supports six different ways of password digest. They are MD2, MD5, SHA-1, SHA-256, SHA-384 and SHA-512. The method of digest can be specified in `web.xml`. They are fully compatible with standard servlet containers such as Apache Tomcat. All users' passwords in database are

[†]Strategy Pattern is a kind of Design Pattern. It defines a family of algorithms, encapsulates each one, and makes them interchangeable. Strategy lets the algorithm vary independently from clients that use it.

hashed password, even site/database managers can not get the plain text of users' passwords. The hashed password is also used in operations of web service interface.

Chapter 8

Web Service Interface

8.1 Introduce web service and metro

“A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.”[9]

–W3C

A web service interface of Web Call Example Application gives a most broad way for clients. Chapter 9 describes a Java ME Client of this web service interface. However, it is definitely not the only client for the web service. As long as a clients implements the interface, it can use the web call service and database.

The web service interface of Web Call Example Application is based on Metro. Metro is a *high-performance, extensible, easy-to-use* web service stack[20]. It is supported by Sun Microsystems. For more detail about Metro, please refer to Metro Homepage at <https://metro.dev.java.net/>.

8.2 SOAP web service

The web service source file is

```
sip.components.webapp.webserviceg.WebCallImpl.
```

List 8.1 shows a fragment of the source code that calls operation in Sip Call Component.

Listing 8.1. Web Service implementation (fragment)

```

...
import sip.components.core.controller.CallController;
...

@WebService(name = "WebCall")
public class WebCallImpl {

    ...
    @WebMethod
    @WebResult(name = "callID")
    public String call(@WebParam(name = "clientA") String clientA,
                      @WebParam(name = "clientB") String clientB,
                      @WebParam(name = "sipAccountName")
                        String sipAccountName,
                      @WebParam(name = "implType") String implType,
                      @WebParam(name = "username") String username,
                      @WebParam(name = "password") String password)
        Throws PersistenceLayerException,
        AuthenticationFailedException {

    ...
    ...
    CallController controller = cf.createAudioCallController();
    controller.setClientA(clientA);
    controller.setClientB(clientB);
    controller.register();
    controller.start();
    ...
}
}

```

language=Java

The implementing class of web service uses the **@WebService** annotation to set the port name, service name and the target namespace. The web service methods use the **WebMethod** to annotate the operation name and actions. The parameter of the method is annotated with **@WebParam**. There are nine web service methods in the implementation Web Service:

- **getUserURIs:** Get a list of user's URIs. The URI (Uniform Resource Identifiers) can be a phone number with international dialing codes or in the form of **sip:USERNAME@HOST:PORT**. Username is the user name registered at host. Host is the domain name or IP address of host. The URIs in this list will be treated as caller/clientA.
- **getContacts:** Get a contact list. The items in the list are instances of

ContactBean. A **ContactBean** has a **contactName** and a **contactURI**.

- **getRecentCalls:** Get a recent calls list. The recent call is just like a call history with limited numbers. In Web Call Example Application, the default size of recent call list is 5. It could be specified in **web.xml** file.
- **getSipAccounts:** Get accounts for sip providers. The register of sip provider's accounts can be done at both desktop and mobile browser views.
- **getImplTypes:** Get the implementation types list of call method. As mentioned in chapter 4, there are five types of implementation. They are Relay Call, Call transfer, Re-invite, SDP swap and Web client.
- **call:** Establish phone calls. In this web service method, the user's phone number and destination phone number will also be stored in database. A Ring Buffer is used to store the recent calls.
- **cancelCall:** Cancel a phone call. To cancel the call, client has to pass the username, password and call ID as the parameter. Server will check if this call session belongs to the user, before cancel the call.
- **getState:** Get a phone call's state. The state could be **INITIALIZED**, **UNREGISTERED**, **REGISTERED**, **DIALING**, **OUTGOING_A**, **OUTGOING_B**, **REGISTER_FAILED**, **ON_CALL**, **CALL_FAILED**, **IDLE**, **ERROR**, **CLOSED** or **REFUSED**.
- **syncContacts:** Synchronize contacts. The method can be used to keep contacts list synchronized on clients and server. If same name with different phone number happens, the number in client will be kept. This method can be used after load/import contact from a third party contact book. e.g. the native contact book in mobile phone. The usage of synchronize contacts will also be explained in chapter 9.

All methods except **getState** and **getImplTypes** require a username and password. The password is in the format of digest. So even if the communication package is intercepted, the third party will not know the plain text of user password. For details about the method please refer the source code in **WebCallImpl**.

8.3 Deploy of Web Service

To deploy web service, the service needs portable artifacts. In Web Call Example Application, the portable artifacts are generated by **art** (annotation processing tool). The **apt** tool is a tool which distributed with **Metro**. It generates portable artifacts used in JAX-WS services.[19] The execution of **apt** is integrated into ant target in **build.xml**.

The web service listener is setted to `WSServletContextListener` in `web.xml`. The url-pattern and implementation is specified at `sun-jaxws.xml`. The two configuration files are located at folder of `WEB-INF` in distributed package(WAR package).

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 Example Application. 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 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)[22].

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.

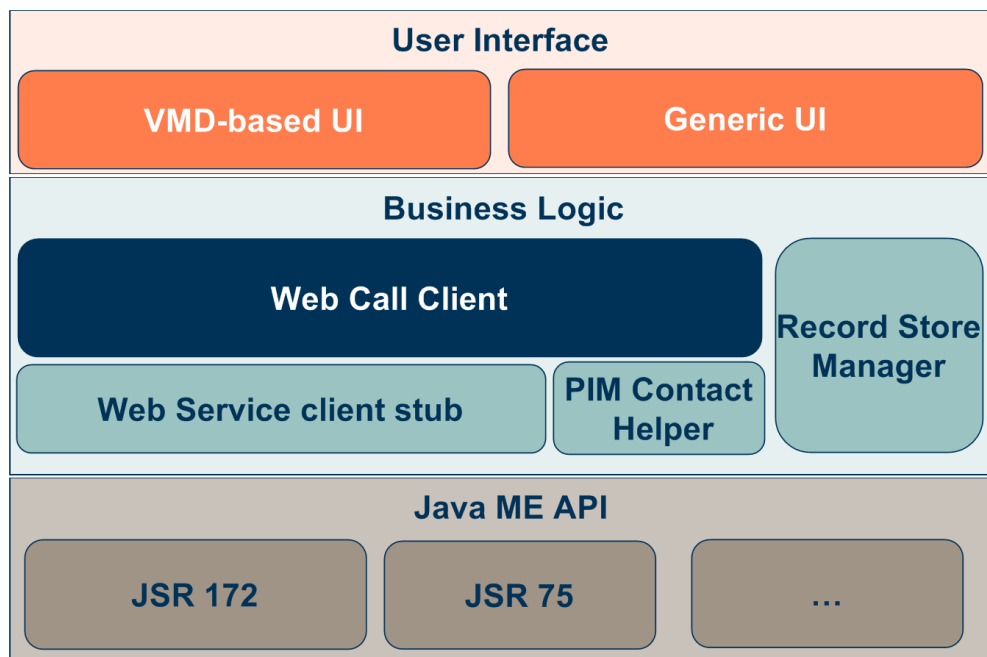


Figure 9.1. The Architecture of Java ME Client

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.

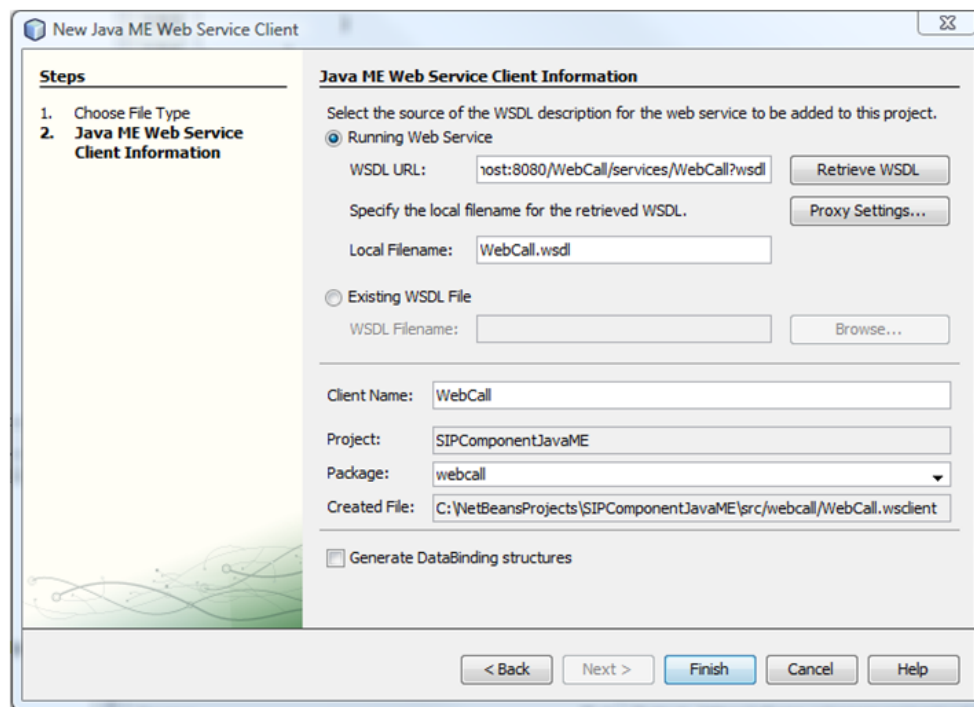


Figure 9.2. Netbeans Java ME Web Service Client Wizard

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

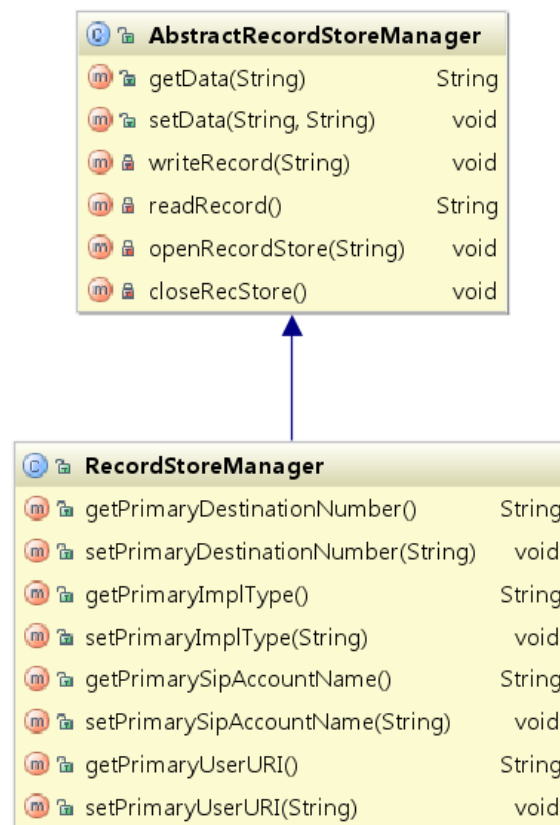


Figure 9.3. The Class Diagram of Record Store Manager

ME applications. To use it, the mobile device must support JSR 75[22]. 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.

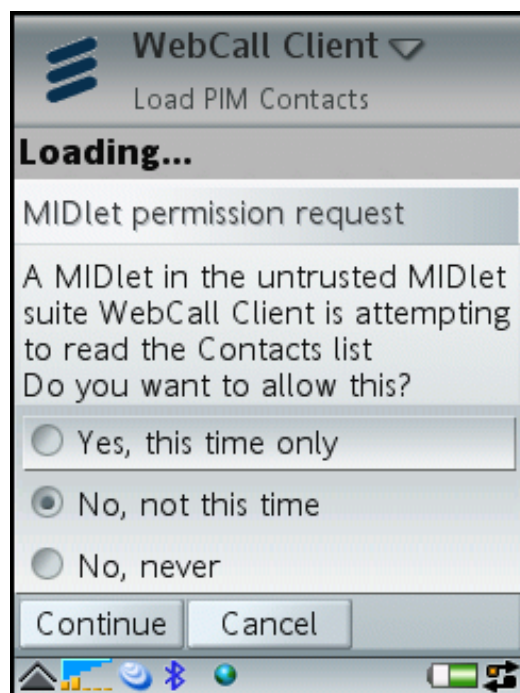


Figure 9.4. Load PIM Contacts

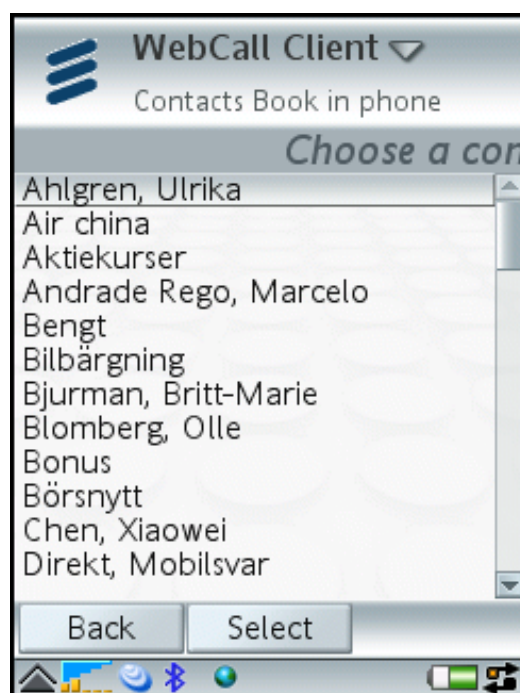


Figure 9.5. PIM Contacts List

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, there 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 7.5.3. 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 chapter 4. 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.

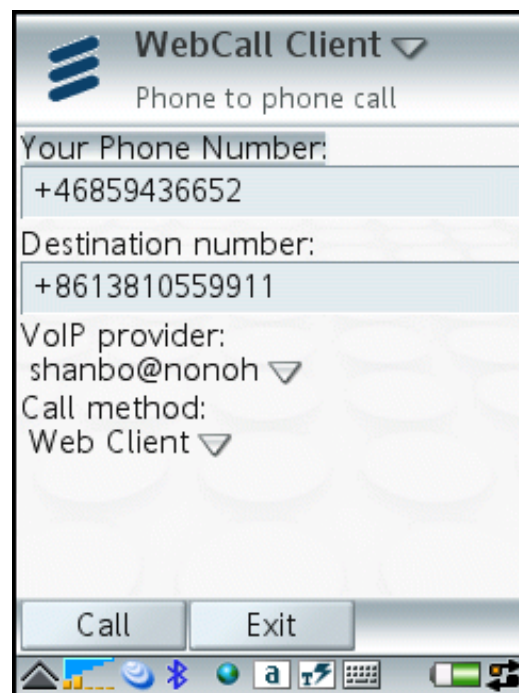


Figure 9.6. VoIP call form of Java ME Client

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 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.

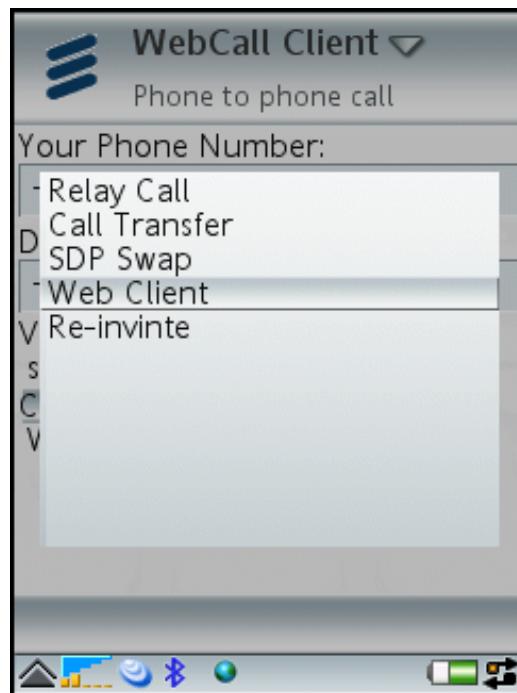


Figure 9.7. Call method drop down list

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.

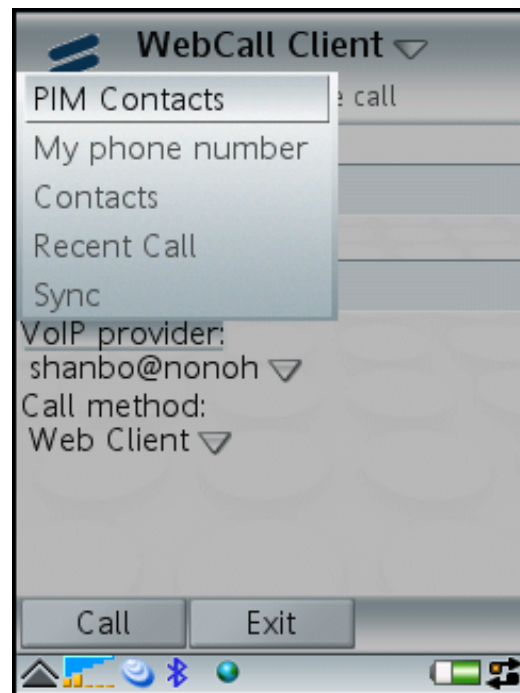


Figure 9.8. Screen menu items of VoIP call form

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

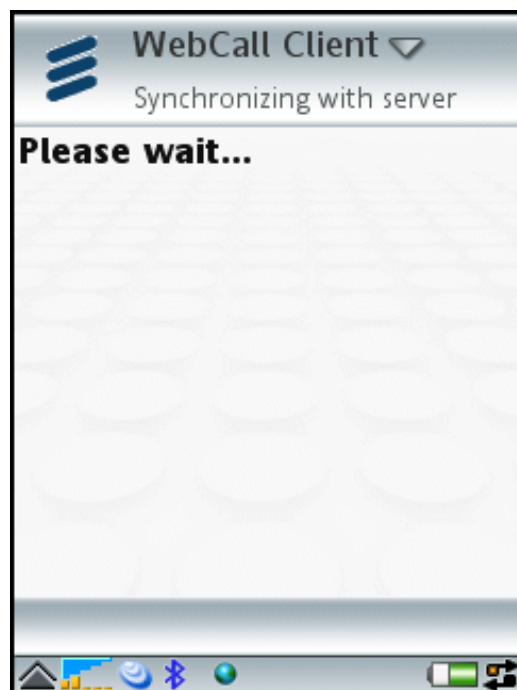


Figure 9.10. Synchronize with server

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

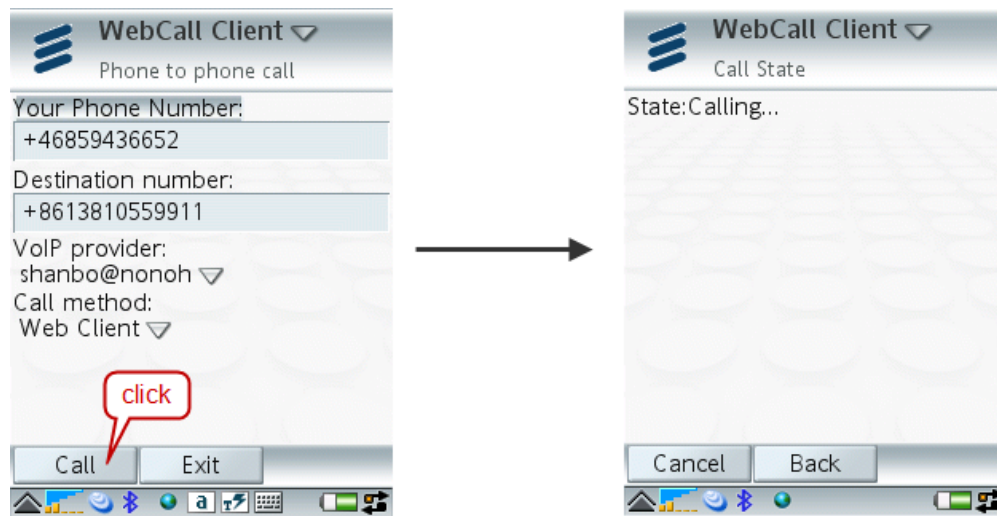


Figure 9.11. Phone call (via VoIP technology) of Java ME client

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** 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 refracted according the package name and Midlet name. All three UIs can be treaded as

*http://shanbohomepage.googlecode.com/svn/trunk/master_thesis/chap09/resources/java_me_working_flow.png

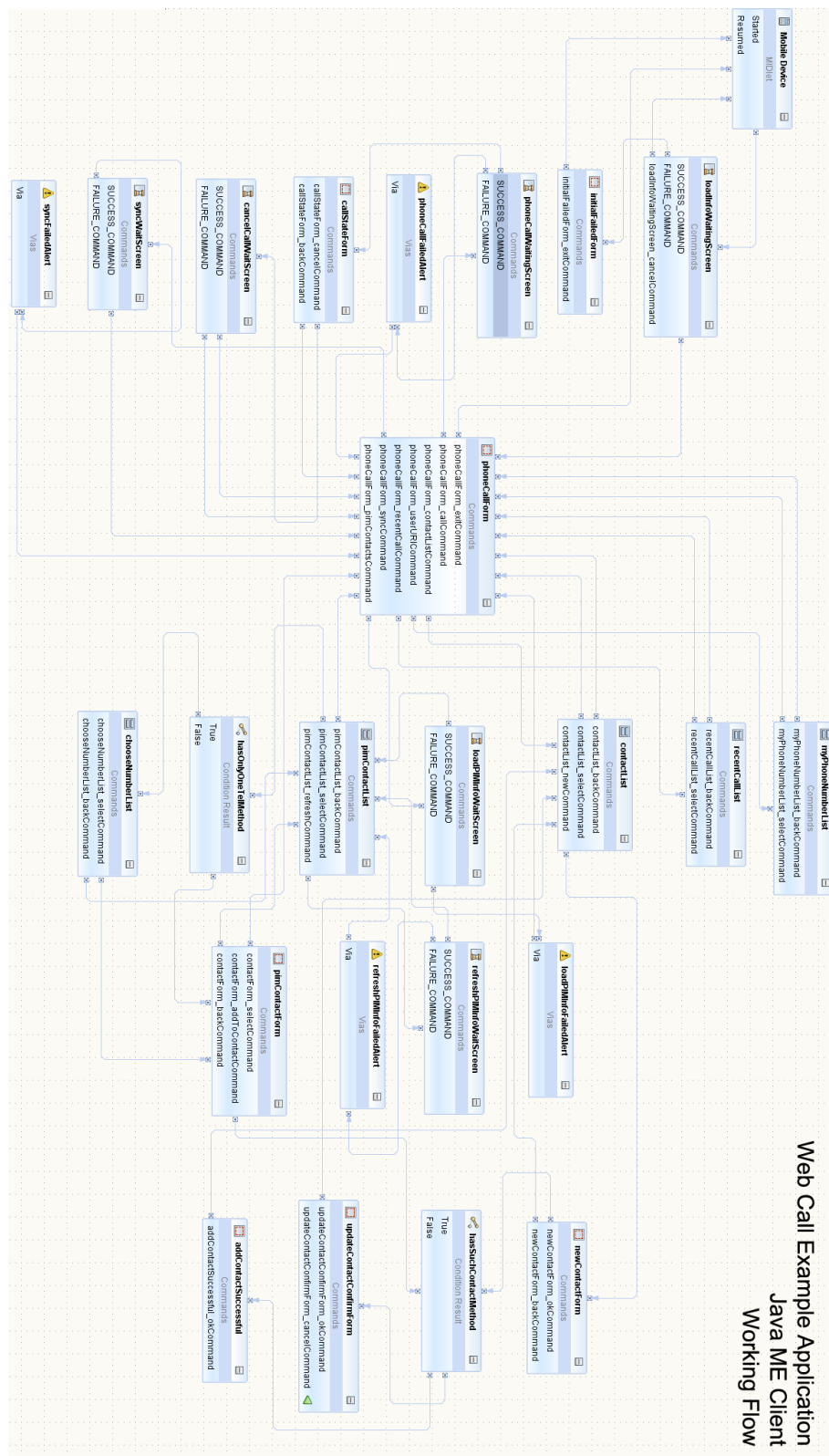


Figure 9.12. Java ME Client Working Flow

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.

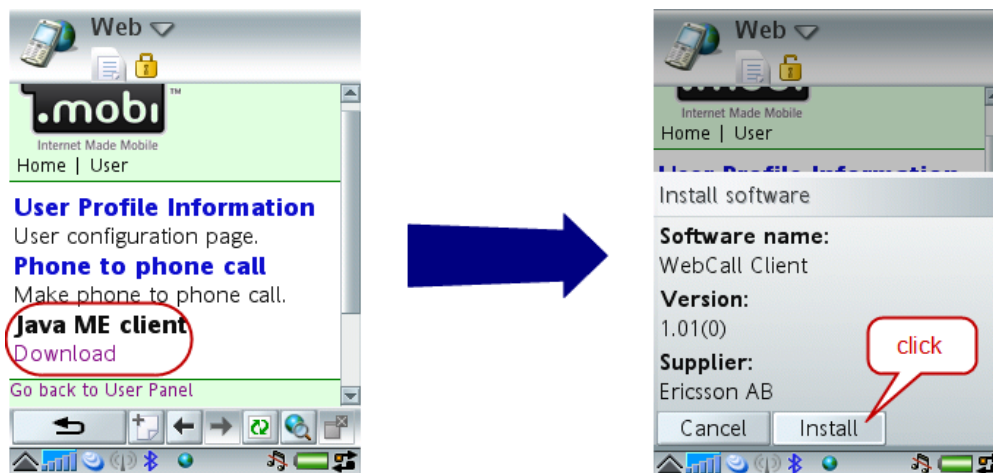


Figure 9.13. Download and install Java ME Client

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 7.10. 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, 29]
- [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. 51]
- [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] Erich Gamma, Richard Helm, Ralph Johnson, and John M. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, November 1994. [cited at p. 30]
- [7] Jesse James Garrett. Ajax: A New Approach to Web Applications.
<http://www.adaptivepath.com/ideas/essays/archives/000385.php>, February 2005. [cited at p. 42]
- [8] Soma Ghosh. J2ME record management store.
<http://www.ibm.com/developerworks/library/wi-rms/>, May 2002. [cited at p. 4]
- [9] Web Services Architecture Working Group. Web Services Glossary.
<http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/>, February 2004. [cited at p. 5, 47]
- [10] 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. 6]
- [11] 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]

- [12] Eric Jendrock, Jennifer Ball, Debbie Carson, Ian Evans, Scott Fordin, and Kim Haase. *Java(TM) EE 5 Tutorial, The 3rd Edition*. Prentice Hall PTR, February 2005. [cited at p. 45]
- [13] 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. 34, 35, 76]
- [14] 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]
- [15] Shanbo Li. Scallope.
<http://scallope.googlecode.com/>, February 2009. [cited at p. 20]
- [16] Richard Marejka. Learning Path: MIDlet Life Cycle.
<http://developers.sun.com/mobility/learn/midp/lifecycle/>, February
 2005. [cited at p. 4]
- [17] Sun Microsystems. Java EE Technology.
<http://java.sun.com/javaee/technologies/>, June 2009. [cited at p. 4]
- [18] Sun Microsystems. Java ME Technology.
<http://java.sun.com/javame/technology/>, June 2009. [cited at p. 4]
- [19] Sun Microsystems. Metro 1.5 FCS – APT.
<https://metro.dev.java.net/1.5/docs/apt.html>, July 2009. [cited at p. 49]
- [20] Sun Microsystems. metro: Home.
<https://metro.dev.java.net/>, July 2009. [cited at p. 47]
- [21] 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]
- [22] 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. 51, 54]
- [23] 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]
- [24] 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, 40]
- [25] 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, 10, 18, 38]

- [26] 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]
- [27] R. Sparks. The Session Initiation Protocol (SIP) Refer Method.
<http://www.ietf.org/rfc/rfc3515.txt>, April 2003. [cited at p. 17]
- [28] Wikipedia. Transport layer security.
http://en.wikipedia.org/wiki/Secure_Sockets_Layer, June 2009. [cited at p. 5]
- [29] Wikipedia. Web 2.0.
http://en.wikipedia.org/wiki/Web_2.0, June 2009. [cited at p. 10]
- [30] Peter Yeung and Pär Johansson. Mobile Front Controller.
http://www.ericsson.com/developer/sub/open/technologies/mobile_browsing/tools/mobile_front_controller, March 2009. [cited at p. 34]

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
Ajax	Asynchronous JavaScript and XML	page 42
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
XHTML	Extensible Hypertext Markup Language	page 35

List of Figures

3.1	Third party call control	12
4.1	The signal and media flow of Relay Call	15
4.2	The signal and media flow of Third Party Call	17
4.3	The signal and media flow of Call Transfer	18
4.4	The signal and media flow of SDP Swap	19
4.5	The signal and media flow of SDP Swap	20
5.1	Use Cases of Web Call Example Application	24
5.2	The Architecture of Web Call Example Application	25
6.1	Five layers architecture	28
6.2	Class diagram of <code>SessionInterface</code>	29
6.3	The Architecture of SIP Call Component	30
6.4	Call Controller class hierarchy diagram	31
6.5	Controller Factory class hierarchy diagram	32
7.1	Welcome page of desktop browser view of web application	33
7.2	An overview of Mobile Front Controller used by a web application on a Java EE web container. (Figure taken from <i>Mobile Front Controller Developer's guide for software version 3.1</i> [13])	34
7.3	The architecture of Mobile Front Controller (Figure taken from <i>Mobile Front Controller Developer's guide for software version 3.1</i> [13])	35
7.4	Welcome page of mobile browser view	36
7.5	User profile information page of desktop browser view	37
7.6	Register VoIP service provider account	38
7.7	User panel desktop browser view	39
7.8	Phone to phone call	39
7.9	User list of administrator panel	40
7.10	Administrator view of user information	41

7.11 Class diagram of CallSessions	42
7.12 EER diagram of database(Diagram generated by MySQL Workbench) .	44
7.13 First user will be administrator	45
9.1 The Architecture of Java ME Client	52
9.2 Netbeans Java ME Web Service Client Wizard	53
9.3 The Class Diagram of Record Store Manager	54
9.4 Load PIM Contacts	55
9.5 PIM Contacts List	55
9.6 VoIP call form of Java ME Client	56
9.7 Call method drop down list	57
9.8 Screen menu items of VoIP call form	58
9.9 Contact list of VoIP call form	59
9.10 Synchronize with server	59
9.11 Phone call (via VoIP technology) of Java ME client	60
9.12 Java ME Client Working Flow	61
9.13 Download and install Java ME Client	62

List of Tables

4.1 Comparison of Call Method	22
---	----