VOIP API

Developer's Guide

E-Science Corporation

Document Build Status

Document Version	Author	Date	Comments
1.0	Alfred Ocon	Mar 17, 2009	Initial document build.
1.0.1	Chris Dacuya	Mar 18, 2009	Updated section 2.13.4 and 2.13.6

Table of Contents

1 About this document

1.1 Audience

Target readers of this document: ***VOIP API Developer's Guide*** are the developers that will use VOIP API. This document declares the design components of VOIP API as well as the dependent systems (if any) that it requires. Functional components are limited to those of the VOIP API system only.

1.2 Structure of this document

Chapter 1, *About This Document*, describes the proper audience this document is created for.

Chapter 2, *General Information*, provides a description on the basic rules the VOIP API application adheres to.

2 General Information

2.1 System Description

The SMS / MMS API project involves the development of an SMS, MMS, LBS API that will be used by developers when they build solution for Globelabs. Aside from the development of the API's, this project will also deliver an application container where developers can optionally store their programs.

2.2 Term Definition

These are the terms used in the manual.

TERM	DEFINITION
API	Application Programming Interface
CLIENT	Mobile Phones, MMS or SMS sender
COMPILED	A program in binary format.
ESC	E-Science Corporation
MMS	Multimedia Messaging System
MMSC	Multimedia Message Service
SMS	Short Messaging System
SMSC	Short Message Service Center
LBS	Location-based Service
CSP	Customer Service Platform
VOIP	Voice Over IP
TPS	Transaction per second.
UDH	User Data Header

2.3 System Flow

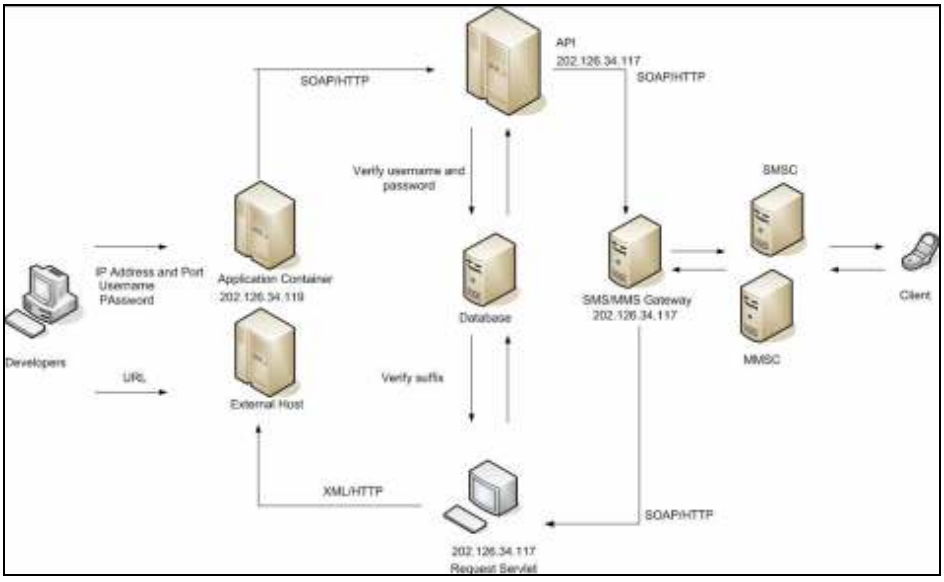


Figure 1

2.4 Introduction

The VOIP Project allows the user to make a call from client, anything that can connect to Sip Server by means of sip packets, to mobile phone and vice versa. It uses standard SIP (RFC3261) packets. It supports G729 codec for audio sound. It also supports recording of successful calls for billing.

The document will discuss the basic operation of the VOIP Project and will guide the readers on how to create a simple client.

2.5 Setup

The VOIP used asterisk as its server. When someone login, asterisk checked if the credentials from sip request matches with the credentials of the user in the database. Upon successful login, the user can start using its services. The user can make and receive call from consented and whitelisted numbers. However, the user might fail to make or receive call if the user already reached the allotted maximum call per day or if the call is made beyond the access period. The call might also end in the middle of conversation when it reached the allotted maximum call duration. Error code should be heard in case of an error with corresponding error message. Every successful dial will be recorded to database (CDR) and only answered calls will be deducted from the maximum call per day.

2.6 VOIP Error Codes

The list of error codes and its description are enumerated below:

Error code	Description
306	The user is disabled.
501	Invalid Mobtel. The user fails to register the number.
701	The target msisdn is blacklisted (consent reply was NO)
702	No Consent.
801	Service unavailable. Request is beyond the access period.
804	Different IP. The IP used during login is different from the IP used in dialing.
805	The user has reached maximum call per day.

To use VOIP service, the user should meet any of the following conditions:

- User should be registered. Upon successful registration, the user will receive an email containing the user account/username, password/pin (for login) and other details.
- User should not be disabled or banned to use the service.
- User should be in profile with VOIP service.
- User should register numbers and asked consent.

2.7 Basic Sip Overview

SIP is an application-layer control protocol that handles the setup, modification, and tear-down of multimedia sessions. SIP is used in combination with other protocols to describe the session characteristics to potential session participants. SIP is based on a request and response transaction model similar to HTTP. Each transaction consists of a request that invokes a particular method or a function on the server and at least one response.

2.7.1 SIP Entities

SIP is generally considered to be a client–server protocol. It has two classes of entities:

Class	Description
Client	A client is an application program that sends a SIP request.
Server	A server generally responds to a request sent by a client.

2.7.2 Different roles of a SIP server

SIP servers have different roles, such as:

Role	Description
Proxy server.	A SIP proxy works like an HTTP proxy server. When a client sends requests to the proxy, the proxy either handles them or forwards them to other servers.
Redirect server.	A SIP redirect server accepts a SIP request and conveys to the originating client the way to route the call.
Registrar server.	A SIP registrar server accepts registration requests and maps a client's address to a user's sign-in name, or SIP URI. Typically, a registrar is combined with a proxy or redirect server.

2.7.3 SIP structure

- Start line
- Message Header
- Message Body

2.7.3.1 SIP request

A SIP request consists of a method token, a request URI, and the SIP version. A method token is used to identify the request. The request URI is the address of the device where the request is being sent.

The original SIP RFC 3261 defines six methods, which are used for different types of requests. The following table describes these methods:

Method Name	Description
INVITE	Initiates a session. This method includes information about the calling and called users and the type of media that is to be exchanged.
ACK	Sent by the client who sends the INVITE. ACK is sent to confirm that the session is established. Media can then be exchanged.
BYE	Terminates a session. This method can be sent by either user.
CANCEL	Terminates a pending request, such as an outstanding INVITE. After a session is established, a BYE method needs to be used to terminate the session.
OPTIONS	Queries the capabilities of the server or other devices. It can be used to check media capabilities before issuing an INVITE.
REGISTER	Used by a client to login and register its address with a SIP registrar server.

2.7.3.2 SIP method extensions

A number of extensions and enhancements have been made to the original SIP RFC 2543. This includes the addition of the following new methods to SIP, which can be used for event notification, instant messaging and call control:

Method Name	Description
SUBSCRIBE	The SUBSCRIBE method enables a user to subscribe to certain events. This means that the user should be informed when such events occur.
NOTIFY	The NOTIFY method is used to inform the user that a subscribed event has occurred.
MESSAGE	SIP can also be used for Instant Messaging. A user sends an instant message to another user by sending a request that includes the MESSAGE method. This request carries the actual text in a body of a SIP packet.
INFO	The INFO method is used for transferring information during a session, such as user activity.
SERVICE	The SERVICE method can carry a Simple Object Access Protocol (SOAP) message as its payload.
NEGOTIATE	The NEGOTIATE method is used to negotiate various kinds of parameters, such as security mechanisms and algorithms.
REFER	A REFER request enables the sender of the request to instruct the receiver to contact a third party using the contact details provided in the request. Call Transfer is a commonly used application of the REFER method.

```
REGISTER sip:202.126.34.117 SIP/2.0
v: SIP/2.0/UDP
202.124.146.34:5060;rport;branch=z9hG4bKPje9e2dcdd0901457aaec
adea67d07a754
Max-Forwards: 70
f:
<sip:cfaa2z@202.126.34.117>;tag=bc4f169970ee4ff8a32f3c38eea5b
e09
t: <sip:cfaa2z@202.126.34.117>
i: bb20939340b74ea49322d8fd80e23a7a
CSeq: 34995 REGISTER
m: <sip:cfaa2z@202.124.146.34:5060>
Expires: 300
Authorization: Digest username="cfaa2z", realm="globelabs",
nonce="48c11928", uri="sip:202.126.34.117",
response="3ac56939e43ba6b59d81ebfb37ff9831", algorithm=MD5
l: 0
```

2.7.3.3 SIP response

A SIP response contains a status code, which is a three-digit number that indicates the outcome of the request. The response also contains a reason phrase, which provides a textual description of the outcome of the request. The reason code is interpreted and acted upon by the client software. The reason phrase helps the user understand the response.

Status codes defined in SIP have values between 100 and 699 and the first digit of the reason code indicates the response class.

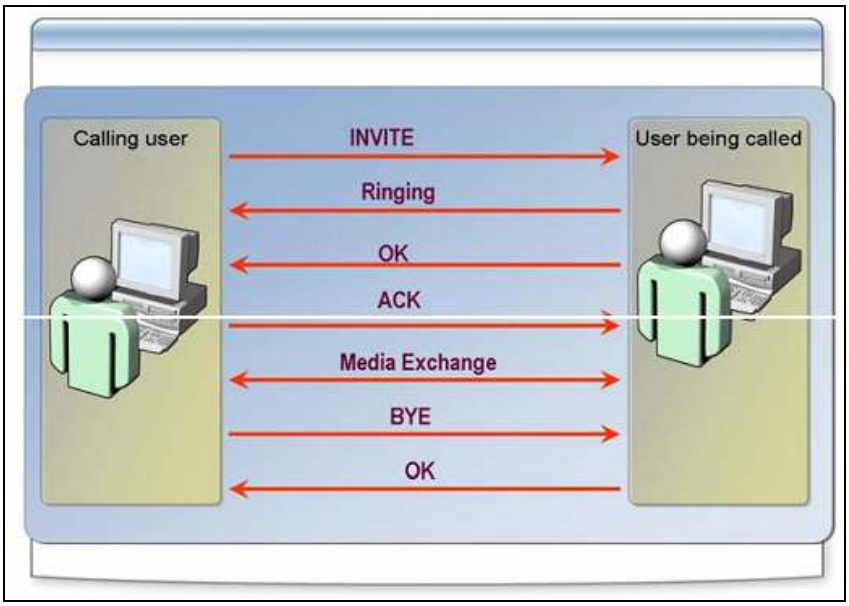
The following table describes the different classes in SIP:

Class Name	Description
1xx: Provisional	Request received, continuing to process the request.
2xx: Success	Action was successfully received, understood, and accepted.
3xx: Redirection	Further action needs to be taken to complete the request.
4xx: Client Error	Request contains bad syntax or cannot be fulfilled at this server.
5xx: Server Error	Server failed to fulfill a valid request.
6xx: Global Failure	Request cannot be fulfilled at any server. This is a new class defined for SIP.

2.7.3.4 Sample SIP message

```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP
202.124.146.34:5060;rport=5060;received=202.124.146.34;branch
=z9hG4bKPj8bd31e8e4c8e4dddb637a2c5263fbc31
From:
<sip:cfaa2z@202.126.34.117>;tag=bc4f169970ee4ff8a32f3c38eea5b
e09
To: <sip:cfaa2z@202.126.34.117>
Call-ID: bb20939340b74ea49322d8fd80e23a7a
CSeq: 34994 REGISTER
User-Agent: eScience
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE,
NOTIFY
Supported: replaces
Contact: <sip:cfaa2z@202.126.34.117>
Content-Length: 0
```

2.7.3.5 Basic SIP Session Establishment



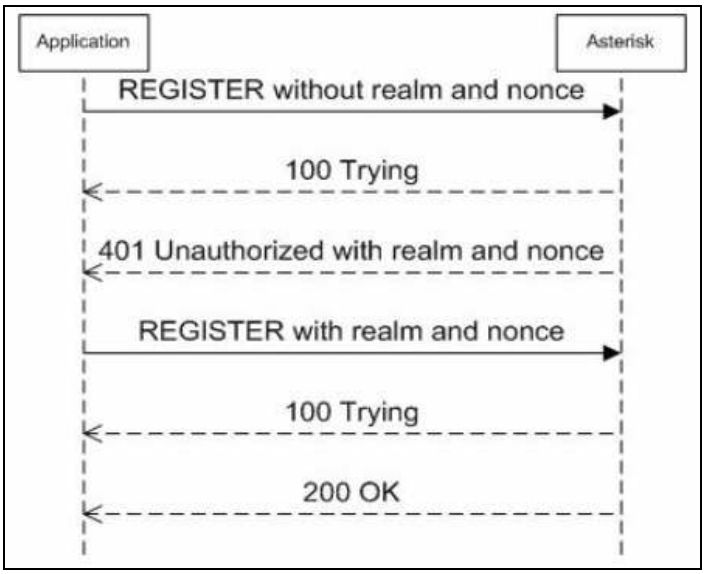
2.8 Basic VOIP

VoIP is a technology that allows us to make phone calls over a broadband Internet connection as opposed to traditional analog phone lines. In simple terms it makes our computer or laptop as telephone.

VoIP technology uses packet switching. It opens connection just long enough to send bits of useful data called packet from one system to another.

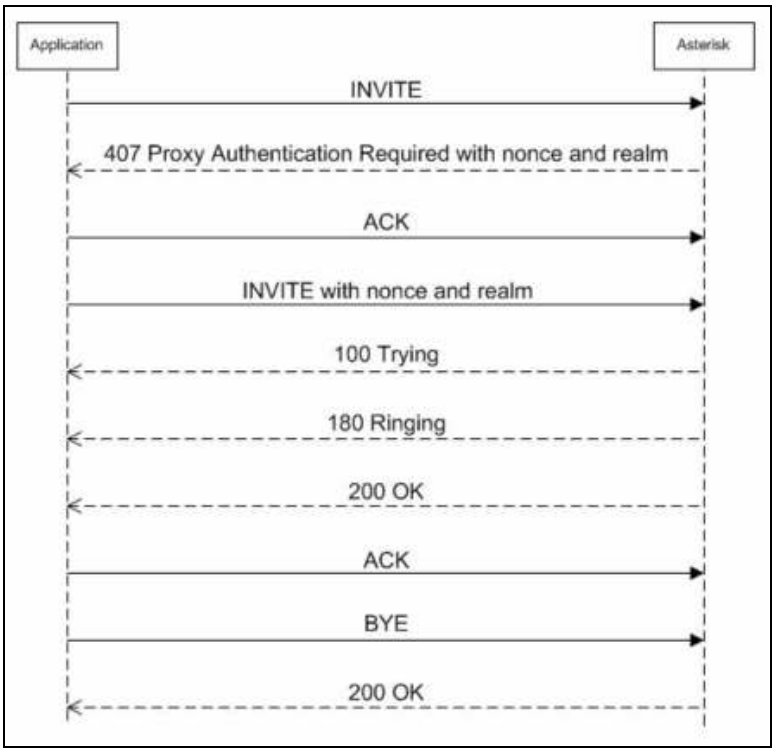
The sample client uses SIP packets to register, unregister and make calls. The sequence of packets for registration, unregistration, making and receiving call are provided below.

2.8.1 Registration and Unregistration



Note: Expires value during registration is 300 and 0 during unregistration. 200 OK may change with error status packets such as 403 Forbidden (wrong password), 404 Not Found (incorrect username) etc.

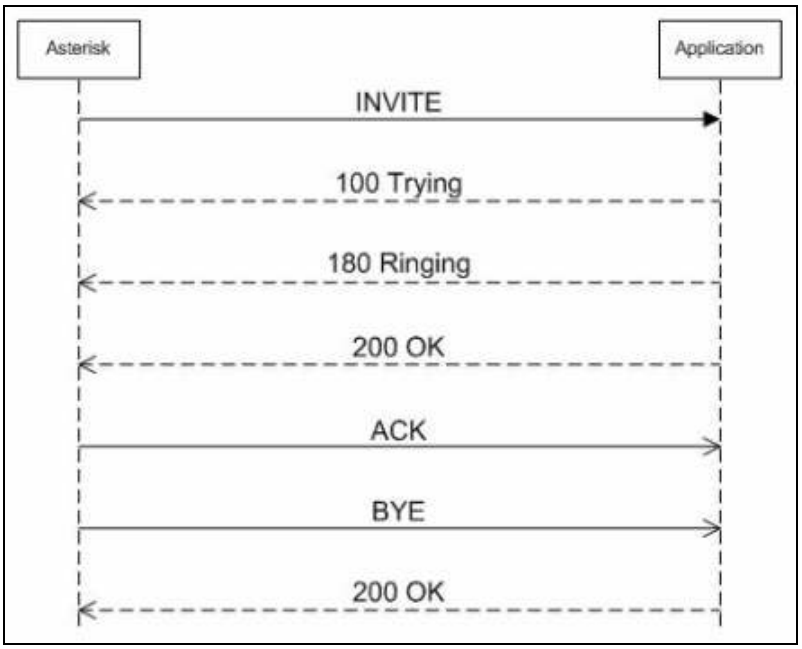
2.8.2 Making a call



Note: 180 Ringing and below packets may change depending on the status of call. It may change to 503 Service Unavailable and stop in ACK. In case validation error, 603 Decline will be issued by the asterisk server.

The BYE part signals the call hang-up or end of call. It may come from any party.

2.8.3 Receiving a call



Note: In this case, no authentication is needed since the asterisk server is the one requesting. 200 OK may change depending on the application response (hang-up).

If in case the application is unable to response the asterisk server will time-out and cancel its request.

2.9 Events

Identification of events is provided by three pieces of information:

- Request URI
- Event Type
- Message body (optional)

The application should implement callbacks to catch the event results. The examples of handling the events in program are given below.

2.9.1 Sample Registration Event

```
Void on_reg_state(int  acc_id)    {

       acc_info accinfo;

               // Parse the packet by providing the acc_id and save
it to struct acc_info
       acc_get_info(acc_id, &accinfo);

       switch (accinfo.status) {
       case 403:
           // STATEMENT FOR WRONG PASSWORD
           System::Windows::Forms::MessageBox::Show("Incorrect
   password!");
           break;
       case 404:
           // STATEMENT FOR INCORRECT USERNAME
           System::Windows::Forms::MessageBox::Show("Incorrect
   username!");
           break;
       default:
           // OTHER ERROR
           System::Windows::Forms::MessageBox::Show("Failed to
   connect!");
           break;
       }

}
```

2.9.2 Sample Outgoing Call Event

```
Void on_call_state(int call_id, event *e)      {

       call_info callinfo;

                // Parse the packet by providing the call_id and
save it to struct call_info
       call_get_info(call_id, &callinfo);


System::Windows::Forms::MessageBox::Show(callinfo.msg);
}
```

2.9.3 Sample Incoming Call Event

```
void on_incoming-call(int call_id, event *e)   {

       call_info callinfo;

                // Parse the packet by providing the call_id and
save it to struct call_info
       call_get_info(call_id, &callinfo);


System::Windows::Forms::MessageBox::Show(callinfo.from + " is
calling");
}
```

2.10 XML-API

2.10.1 Application API IP Address and WSDL

http://202.126.34.117:1881/axis2/services/Platform?wsdl

2.10.2 Get Consent

Before a call is made to a globe mobile number, consent must first be acquired, to this, below are the details on how to send consent request:

2.10.2.1 getConsent Request

```
POST /axis2/services/Platform?wsdl HTTP/1.1
TE: deflate,gzip;q=0.3
Connection: TE, close
Accept: text/xml
Accept: multipart/*
Accept: application/soap
Host: 202.126.34.117:1881
User-Agent: SOAP::Lite/Perl/0.69
Content-Length: 598
Content-Type: text/xml; charset=utf-8
SOAPAction: "http://ESCPlatform/xsd#getConsent"

<?xml version="1.0" encoding="UTF-8"?><soap:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
soap:encodingStyle="http://schemas.xmlsoap.org/soap/encodin
g/"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"><soa
p:Body><getConsent xmlns="http://ESCPlatform/xsd"><apiType
xsi:type="xsd:string">voice</apiType><uName
xsi:type="xsd:string">escience</uName><uPin
xsi:type="xsd:int">9483</uPin><MSISDN
xsi:type="xsd:long">09178544256</MSISDN></getConsent></soap
:Body></soap:Envelope>
```

2.10.2.2 Request Fields

The following are the required nodes for getConsent:

Nodes	Description
apiType	the node which specifies if the request is for lbs or voip. This field must contain the values: ‘voice’ – for voip and ‘location’ for lbs.
uName	the node which specifies the username of the developer.
uPin	the node which specifies the pin of the developer.
MSISDN	the node which specifies the target mobile number for consent.

2.10.2.3 getConsent Response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/xml;charset=utf-8
Date: Tue, 17 Mar 2009 08:34:15 GMT
Connection: close

<?xml version='1.0' encoding='utf-8'?><soapenv:Envelope
xmlns:soapenv=
"http://schemas.xmlsoap.org/soap/envelope/"><soapenv:Body><
ns:getConsentResponse
xmlns:ns="http://ESCPlatform/xsd"><ns:ConsentReturn><ns:ret
urn>203</ns:return>
<ns:tranId>237300010917854425620090317040315 88</ns:tranId><
/ns:ConsentReturn>
</ns:getConsentResponse></soapenv:Body></soapenv:Envelope>
```

2.10.2.4 Response Fields

The following are the nodes for getConsent response:

Nodes	Description
return	specifies the status code of the consent request. Please refer to section 2.13.7 for status code reference.
tranId	the generated transaction id of the request.

2.10.2.5 XML-API responses

ERROR	DESCRIPTION	SMS	MMS	LBS	VOIP
101	Bad Request Message				
201	SMS accepted for delivery				
202	MMS accepted for delivery				
203	LBS accepted for delivery				
301	User is not allowed to access this service				
302	User exceeded daily cap				
303	Invalid message length				
304	Maximum Number of simultaneous connections reached				
305	Invalid login credentials				
306	User is disabled				
307	Invalid target MSISDN				
401	SMS sending failed				
402	MMS sending failed				
403	Subscriber not found (covers				

	the error code 401 coming from CSP)				
407	Subscriber is not authorized to avail of the service. Do not retry.				
409	Subscriber consent is required. Please initiate a request.				
418	Subscriber privacy is set to ON. Do not retry.				
419	Unknown subscriber (GMLC-4 and GMLC-5)				
420	Timeout while locating subscriber (GMLC-500)				
421	Unable to locate subscriber (GMLC-501, GMLC-6)				
501	Invalid Mobtel				
502	Invalid Display type				
503	Invalid mwi				
504	Invalid coding				
505	Empty value given in required argument				
506	Badly formed XML in SOAP request				
507	Argument given too large				
508	Invalid file size				
509	Malformed SMIL URL				
510	LBS Platform encountered an error while processing the request.				
601	Illegal character found in UDH				
602	Maximum location service query reached				
603	There is a pending consent request for this subscriber				
604	Get consent timeout				
605	There is a pending locate request for this number				
700	Whitelisted				
701	Blacklisted				
702	No Consent				
703	Valid Get Location				
704	Valid Get Consent				
705	Valid Test Number				
706	Consent Timeout				

707	Location TImeout				
708	CSP Timeout Error				
709	Location Failure				
801	Request is within the inaccessibility period				
802	Sending to queue failed				
803	Maximum Global Counter reached				
804	IP error				
805	Max call per day				