

begin body

**TITLE**

AUTHOR  
Version  
CREATEDATE



# Table of Contents

Table of contents

## Foreword

### Foreword

**android-ngn-stack** is a [NGN](#) (Next Generation Network) stack for Android 2.x (or later) devices.

The Stack is based on [doubango](#) framework. [doubango](#) is the world's most advanced open source 3GPP IMS/RCS framework for both embedded and desktop systems.

The main purpose is to provide an open source stack for the developers to build their own VoIP applications.

This framework offers a unique set of features ranging from audio/video calls, content sharing, messaging, conferencing, enhanced address book to social presence. All these features are implemented in accordance with the standards: GSMA RCS, 3GPP IMS or VoLTE.

## Introduction

This document has been written by us (Doubango Telecom) to help developers to quickly create innovative multimedia applications for the Android OS. If you are a developer and is looking for the best way to develop a NGN (VoIP, Messaging, Video Conferencing, ...) or rich application for Android then your are at the right place.

If you want to get help or have some feedbacks then please visit our website: <http://code.google.com/p/imsdroid/>

### Doubango Solution

**android-ngn-stack** is part of Doubango Solution which include many components such as:

#### Client-side components

- 1 [Boghe](#): IMS/RCS Client for Windows
- 2 [IMSDroid](#): IMS/RCS Client for Android using **android-ngn-stack**
- 3 [iDoub](#): IMS/RCS Client for iOS (iPhone, iPad and iPod Touch)

#### Server-side components

- 4 [OpenVCS](#): OpenVCS stands for Open Source Video Conferencing Server and is used to manage Multipoint Control Units (MCU). Each MCU (a.k.a Bridge) can handle up to 64 participants
- 5 [Flash2IMS](#): Adobe® Flash® to SIP/IMS Gateway.

## Highlights

- 6 SIP(RFC 3261, 3GPP TS 24.229 Rel-9)
- 7 TCP and UDP over IPv4 or IPv6
- 8 Signaling Compression, SigComp(RFC 3320, 3485, 4077, 4464, 4465, 4896, 5049, 5112 and 1951)
- 9
- 10 Enhanced Address Book (XCAP storage, authorizations, presence, ...)
- 11 GSMA Rich Communication Suite release 3
- 12 Partial supports for One Voice Profile V1.0.0 (GSMA VoLTE)
- 13 Partial supports for MMTel UNI (used by GSMA RCS and GSMA VoLTE)
- 14
- 15 IMS-AKA registration (both AKA-v1 and AKA-v2), Digest MD5, Basic
- 16 3GPP Early IMS Security (3GPP TS 33.978)
- 17 Proxy-CSCF discovery using DNS NAPTR+SRV
- 18 Private extension headers for 3GPP
- 19 Service Route discovery
- 20 Subscription to reg event package (Honoring network initiated (re/de/un)-registration events)
- 21
- 22 3GPP SMS Over IP (3GPP TS 23.038, 24.040, 24.011, 24.341 and 24.451)
- 23 Voice Call (G729AB1, AMR-NB, iLBC, GSM, PCMA, PCMU, Speex-NB)
- 24 Video Call (H264, MP4V-ES, Theora, H.263, H.263-1998, H.261)
- 25 DTMF (RFC 4733)
- 26 QoS negotiation using Preconditions (RFC 3312, 4032 and 5027)
- 27 SIP Session Timers (RFC 4028)
- 28 Provisional Response Acknowledgments (PRACK)
- 29 Communication Hold (3GPP TS 24.610)
- 30 Message Waiting Indication (3GPP TS 24.606)
- 31 Calling E.164 numbers by using ENUM protocol (RFC 3761)
- 32 NAT Traversal using STUN2 (RFC 5389) with possibilities to automatically discover the server by using DNS SRV (TURN already implemented and ICE is under tests)
- 33
- 34 One2One and Group Chat
- 35 File Transfer and Content sharing
- 36

## Setting up NGN project

This section explain how to setup a NGN project using Eclipse.

### Checking out the source code

To check out the source code of the NGN library you will need a SVN client.

Use this command to anonymously check out the last project source:

```
svn checkout http://imsdroid.googlecode.com/svn imsdroid
```

The source code of the library is under:

```
imsdroid/branches/2.0/android-ngn-stack
```

## Importing the NGN project into Eclipse

The NGN project is the Next Generation Network library.

- 37 Open eclipse
- 38 Go to File -> Import -> General -> Existing Project into workspace
- 39 Select **android-ngn-stack** folder and click **Finish**

## Creating you first NGN application using Eclipse

- 40 Open Eclipse and select File -> New -> Android Project
- 41 From the next window ("New Android Project" ) fill the text fields like this:
- 42
- 43 Project name: **myFirstApp**
- 44 Location: < set any path >
- 45 Build Target: **Android 2.0** (at least)
- 46 Application name: **myFirstApp**
- 47 Package name: **org.doubango.test**
- 48 Check "**Create Activity**" and name it "**Main**"
- 49
- 50
- 51 Click on Finish to create the project
- 52 From the Eclipse package explorer, right click on **myFirstApp** and select "**Properties**" then "**Android**" from the left
- 53
- 54
- 55 From the properties window, select "**Add**" button then select **android-ngn-stack** from the list of the available libraries
- 56
- 57
- 58 Select "**Java Compiler**" from the left and change the version from 1.5 to 1.6
- 59
- 60
- 61 Select "**Java Build Path**" from the left, then "**Libraries**"
- 62
- 63
- 64 From "Java Build Path 1/2", select "**Add JARs...**" then **android-ngn-stack/libs/simple-xml-2.3.4.jar** , then "**OK**" to close the window
- 65
- 66
- 67 Click on "**OK**" to close the window

## Setting up Android Permissions

In order to use the framework you must enable some user-permission in your Android manifest.

Open **myFirstApp/AndroidManifest.xml** , then add this:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />

<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
```

```

<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />

<uses-permission android:name="android.permission.WRITE_SETTINGS" />
<uses-permission android:name="android.permission.DISABLE_KEYGUARD" />
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<uses-permission android:name="android.permission.WRITE_CONTACTS"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.PROCESS_OUTGOING_CALLS" />
<uses-permission android:name="android.permission.CALL_PHONE" />
<uses-permission android:name="android.permission.RAISED_THREAD_PRIORITY"/>
... just before
</manifest>

```

## Declaring your app as NGN

Declaring your app as NGN is recommended if you are programming at **high** level.

- 68 From the Eclipse package explorer, open **AndroidManifest.xml** and select **Application** tab from below
- 69 Click on **browse** (on the right of **Name** ) then, select "**NgnApplication**" from the list
- 70
- 71

## Architecture

The stack offers three levels of programming: **Low** , **Medium** and **High** .

**Before building and running your project, you should take a look at the section [explaining how to setup a NGN project](#).** **Low level**

This level allow you to directly have access to doubango functions through JNI. This level is the most flexible one but is out of scope because it's too difficult to manage.

All functions used in this level are in one single package: **org.doubango.tinyWRAP**

For example, the code below shows how to register to a SIP/IMS server:

```

final String realm = "sip:doubango.org";
final String privateIdentity = "001";
final String publicIdentity = "sip:001@doubango.org";
final String password = "my secret";
final String proxyHost = "192.168.0.1";
RegistrationSession registrationSession;
// Sip Callback
final SipCallback callback = new SipCallback(){
    @Override
    public int OnDialogEvent(DialogEvent e) {
        final SipSession sipSession = e.getBaseSession();
        final long sipSessionId = sipSession.getId();
        final short code = e.getCode();
        switch (code){
            case
tinyWRAPConstants.tsip_event_code_dialog_connecting:
                if(registrationSession !=
null && registrationSession.getId() == sipSessionId){
                    //
Registration in progress
                }
                break;
            case
tinyWRAPConstants.tsip_event_code_dialog_connected:
                if(registrationSession !=

```

```

null && registrationSession.getId() == sipSessionId){
                                                                    // You
are registered
                                                                    }
                                                                    break;
                                                                    case
tinyWRAPConstants.tsip_event_code_dialog_terminating:
                                                                    if(registrationSession !=
null && registrationSession.getId() == sipSessionId){
                                                                    // You
are unregistering
                                                                    }
                                                                    break;
                                                                    case
tinyWRAPConstants.tsip_event_code_dialog_terminated:
                                                                    if(registrationSession !=
null && registrationSession.getId() == sipSessionId){
                                                                    // You
are unregistered
                                                                    }
                                                                    break;
                                                                    }
                                                                    return 0;
                                                                    }

@Override
public int OnRegistrationEvent(RegistrationEvent e) {
    // low level events
    return 0;
}

};
// Create the SipStack
SipStack sipStack = new SipStack(callback, realm, privateIdentity, publicIdentity);
// Set Proxy Host and port
sipStack.setProxyCSCF(proxyHost, 5060, "UDP", "IPv4");
// Set password
sipStack.setPassword(password);
if(sipStack.isValid()){
    if(sipStack.start()){
        registrationSession = new RegistrationSession(sipStack);
        registrationSession.setFromUri(publicIdentity);
        // Send SIP register request
        registrationSession.register_();
    }
}
}

```

## Medium level

This level is built on of the **low** level. The main advantage of this level is that it's flexible without being too complicated as all low level functions are wrapped into comprehensive API. For example, if you want to implement a multi-stack (multi-account) application this is the right level.

## High level

This level is built in top of the **low** level and is much easier than the later. The High level is composed of a set of Services managed by a single NGN engine instance. Each service is responsible for a particular task. For example, you have one service for SIP, one for contact management, one for networking etc etc

## NGN Engine

The engine is a black box containing all the services. You must always retrieve the services

through the engine.

You must also start/stop the services through the NGN engine.

The code below shows how to get an instance of the engine:

```
// Gets an instance of the engine. This function will always returns the same instance
// which means that you can call it as many as you want from anywhere in your code
final NgnEngine mEngine = NgnEngine.getInstance();
```

The code below shows how to get some services from the engine:

```
// Gets the configuration service
INgnConfigurationService mConfigurationService = mEngine.getConfigurationService();
// Gets the SIP/IMS service
INgnSipService mSipService = mEngine.getSipService();
// etc etc
@endocde
```

The code below shows how to start/stop the engine.

```
@code
// Starts the engine
mEngine.start();
// Stops the engine
mEngine.stop();
```

Starting/Stopping the engine will start/stop all underlying services.

/\*\* Global object defining the application. You should extends this class in your own Android application.

## Base Service

All NGN services inherits from this class.

## Contact Service

The Contact service is used to retrieve contacts from the native address book.

## HTTP/HTTPS Service

The HTTP/HTTPS service is used to send and retrieve data to/from remote server using HTTP/HTTPS protocol.

## Network Service

The network service is used to manage both WiFi and 3g/4g network connections.

## Sound Service

The sound service is used to play the tones (ringtone, ringback, alert, ...). You have to start the service through the NGN engine before any use.

```
// Gets and instance of the NGN engine
NgnEngine mEngine = NgnEngine.getInstance();
// Plays the ringback tone
mEngine.getSoundService().startRingBackTone();
// Stops the ringback tone
mEngine.getSoundService().stopRingBackTone();
```



## Storage Service

This service is used to manage storage functions.

## Configuration Service

The configuration service is used to store the user preferences. All preferences saved using this service are persistent which means that you can retrieve them when the application/device restarts.

You should never create or start this service by yourself.

An instance of this service could be retrieved like this:

```
final INgnConfigurationService mConfigurationService =  
NgnEngine.getInstance().getConfigurationService();
```

## History Service

This service is used to store/retrieve history event (audio/video, messaging, ...). You should never create or start this service by yourself.

An instance of this service could be retrieved like this:

```
final INgnHistoryService mHistoryService = NgnEngine.getInstance().getHistoryService();
```

## SIP/IMS Service

This service is used to manage the SIP/IMS stack. You should never create or start this service by yourself.

An instance of this service could be retrieved like this:

```
final INgnSipService mSipService = NgnEngine.getInstance().getSipService();
```

## Audio/Video calls

### Making audio call

To get notified about the audio/video call state please see [here](#)

```
final String remoteUri = "+33600000000";  
final String validUri = NgnUriUtils.makeValidSipUri(remoteUri); // sip:  
+33600000000@doubango.org  
NgnAVSession avSession = NgnAVSession.createOutgoingSession(mSipService.getSipStack(),  
NgnMediaType.Audio);  
if(avSession.makeCall(validUri)){  
    Log.d(TAG,"all is ok");  
}  
else{  
    Log.e(TAG,"Failed to place the call");  
}
```

### Making video call

To get notified about the audio/video call state please see [here](#)

```

final String remoteUri = "+33600000000";
final String validUri = NgnUriUtils.makeValidSipUri(remoteUri); // sip:
+33600000000@doubango.org
NgnAVSession avSession = NgnAVSession.createOutgoingSession(mSipService.getSipStack(),
NgnMediaType.AudioVideo);
if(avSession.makeCall(validUri)){
    Log.d(TAG,"all is ok");
}
else{
Log.e(TAG,"Failed to place the call");
}

```

## SMS and Chat

### 3GPP Binary SMS

```

final String SMSC = "sip:+3310000000@doubango.org"; // SMS Center
final String remotePartyUri = "sip:+336000000@doubango.org"; // remote party
final String textToSend = "hello world!";
final NgnMessagingSession imSession =
NgnMessagingSession.createOutgoingSession(mSipService.getSipStack(),
remotePartyUri);
if(!imSession.SendBinaryMessage(textToSend,SMSC)){
    Log.e(TAG,"Failed to send");
}
else{
Log.d(TAG,"Message sent");
}
// release session
NgnMessagingSession.releaseSession(imSession);

```

### Pager Mode IM

```

final String textToSend = "hello world!";
final String remotePartyUri = "sip:+336000000@doubango.org"; // remote party
final NgnMessagingSession imSession =
NgnMessagingSession.createOutgoingSession(mSipService.getSipStack(),
remotePartyUri);
if(!imSession.sendTextMessage(textToSend)){
    Log.e(TAG,"Failed to send");
}
else{
Log.d(TAG,"Message sent");
}
// release session
NgnMessagingSession.releaseSession(imSession);

```

## Listening to events

The SIP/IMS service is responsible for all task related to the SIP protocol (Registration, audio/video calls, Pager mode IM, Presence, ...) and you can subscribe to the event changed in order to get notified when the registration state change, new SIP MESSAGE is received, new incoming audio/video call, ...

All notifications are sent to you in asynchronous way which mean that you don't need to query for them more than once.

## Listening for registration state change

You can listen to the registration state change in order to get notified when you are logged in/out.

```
final TextView mTvInfo = (TextView)findViewById(R.id.textViewInfo);
final BroadcastReceiver mSipBroadCastRecv = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context,
                           Intent intent) {
        final String action =
            intent.getAction();

        // Registration Event
        if (NgnRegistrationEventArgs.ACTION_REGISTRATION_EVENT.equals(action)) {
            NgnRegistrationEventArgs args = intent.getParcelableExtra(NgnRegistrationEventArgs.EXTRA_EMBEDDED);
            if (args == null) {
                Log.e(TAG, "Invalid event args");
                return;
            }
            switch (args.getEventType()) {
                case REGISTRATION_NOK:
                    mTvInfo.setText("Failed to register :(");
                    break;
                case UNREGISTRATION_OK:
                    mTvInfo.setText("You are now unregistered :)");
                    break;
                case REGISTRATION_OK:
                    mTvInfo.setText("You are now registered :)");
                    break;
                case REGISTRATION_INPROGRESS:
                    mTvInfo.setText("Trying to register...");
                    break;
                case UNREGISTRATION_INPROGRESS:
                    mTvInfo.setText("Trying to unregister...");
                    break;
                case UNREGISTRATION_NOK:
                    mTvInfo.setText("Failed to unregister :(");
                    break;
            }
        }
    }
};
final IntentFilter intentFilter = new IntentFilter();
intentFilter.addAction(NgnRegistrationEventArgs.ACTION_REGISTRATION_EVENT);
registerReceiver(mSipBroadCastRecv, intentFilter);
```

## Listening for audio/video call state change

You can listen to the audio/video call state change in order to get notified when the call state change (incoming, incall, outgoing, terminated, ...).

```
final BroadcastReceiver mSipBroadCastRecv = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        InviteState state;
        final String action = intent.getAction();
        if (NgnInviteEventArgs.ACTION_INVITE_EVENT
.equals(action)) {
            NgnInviteEventArgs args =
            intent.getParcelableExtra (NgnEventArgs.EXTRA_EMBEDDED);
            if (args == null) {
                Log.e(TAG
, "Invalid event args");
                return;
            }
            Log.d(TAG, "This is an
event for session number "+args.getSessionId());
            // Retrieve the session
            from the store
            NgnAVSession avSession =
            NgnAVSession.getSession(args.getSessionId());
            if (avSession == null) {
                Log.e(TAG
, "Cannot find session");
                return;
            }
            switch ((state =
            avSession.getState())) {
                case NONE:
                default:
                    break;
                case
                INCOMING:
                    Log.i(TAG, "Incoming call");
                    break;
                case
                INPROGRESS:
                    Log.i(TAG, "Call in progress");
                    break;
                case
                REMOTE_RINGING:
                    Log.i(TAG, "Remote party is ringing");
                    break;
                case
                EARLY_MEDIA:
                    Log.i(TAG, "Early media started");
                    break;
                case
                INCALL:
                    Log.i(TAG, "Call connected");
                    break;
```

```

TERMINATING:
    Log.i(TAG, "Call terminating");
    break;

TERMINATED:
    Log.i(TAG, "Call terminated");
    break;
    }
}
};

```

## Configuration

Before trying to register to the SIP/IMS server you must configure the stack with your credentials.

The configuration service is responsible of this task. **All preferences defined using the configuration service are persistent which means that you can retrieve them when the application/device restarts** . To configure the stack you must get an instance of the configuration service from the engine like this:

```
final INgnConfigurationService mConfigurationService =  
NgnEngine.getInstance().getConfigurationService();
```

## Realm

The **realm** is the name of the domain to authenticate to. It should be a valid SIP URI (e.g. *sip:open-ims.test* or *sip:10.0.0.1* ).

The **realm** is mandatory and should be set before the stack starts. You should never change its value once the stack is started. If the address of the Proxy-CSCF is missing, then the stack will automatically use DNS NAPTR+SRV and/or DHCP mechanisms for dynamic discovery.

The value of the **realm** will be used as domain name for the DNS NAPTR query. For more information about how to set the Proxy-CSCF IP address and port, please refer to section 22.1.8.

```
final String myRealm = "sip:doubango.org";
final boolean bSaveNow = true;
mConfigurationService(ConfigurationEntry.NETWORK_REALM, myRealm, bSaveNow);
```

## IMS Private Identity (IMPI)

The IMS Private Identity (a.k.a **IMPI** ) is a unique identifier assigned to a user (or UE) by the home network. It could be either a SIP URI (e.g. *sip:bob@open-ims.test* ), a tel URI (e.g. *tel:+33100000* ) or any alphanumeric string (e.g. *bob@open-ims.test* or *bob* ). It is used to authenticate the UE (username field in SIP Digest Authorization/Proxy-Authorization header).

In the real world, it should be stored in an UICC (Universal Integrated Circuit Card). For those using this IMS stack as a basic (IETF) SIP stack, the IMPU should coincide with their authentication name.

The **IMPI is mandatory** and should be set before the stack starts. You should never change the

**IMPI** once the stack is started.

```
final String myIMPI = "33446677887";
final boolean bSaveNow = true;
mConfigurationService(ConfigurationEntry.IDENTITY_IMPI, myIMPI, bSaveNow);
```

### IMS Public Identity (IMPU)

As its name says, it's your public visible identifier where you are willing to receive calls or any demands. An IMPU could be either a SIP or tel URI (e.g. *tel:+33100000* or *sip:[bob@open-ims.test](mailto:bob@open-ims.test)*). In IMS world, a user can have multiple IMPUs associated to its unique IMPI.

For those using this IMS stack as basic SIP stack, the IMPU should coincide with their SIP URI (a.k.a SIP address).

The **IMPU is mandatory** and should be set before the stack starts. You should never change the IMPU once the stack is started (instead, change the P-Preferred-Identity if you want to change your default public identifier).

```
final boolean bSaveNow = true;
final String myIMPU = "sip:33446677887@doubango.org";
mConfigurationService(ConfigurationEntry.IDENTITY_IMPU, myIMPU, bSaveNow);
```

### Preferred Identity

As a user has multiple IMPUs, it can for each outgoing request, defines which IMPU to use by setting the preferred identity. The user should check that this IMPU is not barred. An IMPU is barred if it doesn't appear in the associated URIs returned in the 200 OK REGISTER.

By default, the preferred identity is the first URI in the list of the associated identities. If the IMPU used to REGISTER the user is barred, then the stack will use the default URI returned by the SCSCF.

You should never manually set this SIP header (P-Preferred-Identity); it's up to the stack.

### Proxy-CSCF Host address

The Proxy-CSCF Host is the IP address (192.168.0.1) or FQDN (doubango.org) of the SIP registrar.

You should set the Proxy-CSCF address and IP only if required. Dynamic discovery mechanisms (DNS NAPTR and/or DHCPv4/v6) should be used. The code below shows how to set the Proxy-CSCF IP address and Port. If the port is missing, then its default value will be 5060.

```
// Sets IP address
final String proxyHost = "192.168.0.1";
mConfigurationService(ConfigurationEntry.NETWORK_PCSCF_HOST, proxyHost);
// Sets port
final int proxyPort = 5060;
mConfigurationService.putInt(ConfigurationEntry.NETWORK_PCSCF_PORT, proxyPort);
Save changes
mConfigurationService.commit();
```

## Class Index

### Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

org.doubango.ngn.sip.NgnSipSession.ConnectionState.....	16
org.doubango.ngn.services.impl.NgnBaseService.....	22
org.doubango.ngn.services.impl.NgnContactService.....	24
org.doubango.ngn.services.impl.NgnHttpClientService.....	25
org.doubango.ngn.services.impl.NgnNetworkService.....	29
org.doubango.ngn.services.impl.NgnSoundService.....	36
org.doubango.ngn.services.impl.NgnStorageService.....	37
org.doubango.ngn.model.NgnContact.....	22
org.doubango.ngn.NgnEngine.....	25
org.doubango.ngn.events.NgnEventArgs.....	25
org.doubango.ngn.events.NgnInviteEventArgs.....	25
org.doubango.ngn.events.NgnStackEventArgs.....	37
org.doubango.ngn.events.NgnStringEventArgs.....	38
org.doubango.ngn.events.NgnInviteEventTypes.....	26
org.doubango.ngn.NgnNativeService.....	29
org.doubango.ngn.sip.NgnPresenceStatus.....	29
org.doubango.ngn.media.NgnProxyPlugin.....	30
org.doubango.ngn.media.NgnProxyAudioConsumer.....	30
org.doubango.ngn.media.NgnProxyAudioProducer.....	30
org.doubango.ngn.media.NgnProxyVideoProducer.....	31
org.doubango.ngn.media.NgnProxyVideoConsumerGL.....	30
org.doubango.ngn.media.NgnProxyVideoConsumerSV.....	31
org.doubango.ngn.sip.NgnSipSession.....	32
org.doubango.ngn.sip.NgnInviteSession.....	26
org.doubango.ngn.sip.NgnAVSession.....	16
org.doubango.ngn.sip.NgnMsrpSession.....	28
org.doubango.ngn.sip.NgnMessagingSession.....	27
org.doubango.ngn.sip.NgnRegistrationSession.....	31
org.doubango.ngn.sip.NgnSipStack.....	36
org.doubango.ngn.events.NgnStackEventTypes.....	37

## Class Index

### Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#"><u>org.doubango.ngn.sip.NgnSipSession.ConnectionState</u></a> .....	16
<a href="#"><u>org.doubango.ngn.sip.NgnAVSession</u></a> .....	16
<a href="#"><u>org.doubango.ngn.services.impl.NgnBaseService</u></a> .....	22
<a href="#"><u>org.doubango.ngn.model.NgnContact</u></a> .....	22
<a href="#"><u>org.doubango.ngn.services.impl.NgnContactService</u></a> .....	24

<a href="#">org.doubango.ngn.NgnEngine</a>	25
<a href="#">org.doubango.ngn.events.NgnEventArgs</a>	25
<a href="#">org.doubango.ngn.services.impl.NgnHttpClientService</a>	25
<a href="#">org.doubango.ngn.events.NgnInviteEventArgs</a>	25
<a href="#">org.doubango.ngn.events.NgnInviteEventTypes</a>	26
<a href="#">org.doubango.ngn.sip.NgnInviteSession</a>	26
<a href="#">org.doubango.ngn.sip.NgnMessagingSession</a>	27
<a href="#">org.doubango.ngn.sip.NgnMsrpSession</a>	28
<a href="#">org.doubango.ngn.NgnNativeService</a>	29
<a href="#">org.doubango.ngn.services.impl.NgnNetworkService</a>	29
<a href="#">org.doubango.ngn.sip.NgnPresenceStatus</a>	29
<a href="#">org.doubango.ngn.media.NgnProxyAudioConsumer</a>	30
<a href="#">org.doubango.ngn.media.NgnProxyAudioProducer</a>	30
<a href="#">org.doubango.ngn.media.NgnProxyPlugin</a>	30
<a href="#">org.doubango.ngn.media.NgnProxyVideoConsumerGL</a>	30
<a href="#">org.doubango.ngn.media.NgnProxyVideoConsumerSV</a>	31
<a href="#">org.doubango.ngn.media.NgnProxyVideoProducer</a>	31
<a href="#">org.doubango.ngn.sip.NgnRegistrationSession</a>	31
<a href="#">org.doubango.ngn.sip.NgnSipSession</a>	32
<a href="#">org.doubango.ngn.sip.NgnSipStack</a>	36
<a href="#">org.doubango.ngn.services.impl.NgnSoundService</a>	36
<a href="#">org.doubango.ngn.events.NgnStackEventArgs</a>	37
<a href="#">org.doubango.ngn.events.NgnStackEventTypes</a>	37
<a href="#">org.doubango.ngn.services.impl.NgnStorageService</a>	37
<a href="#">org.doubango.ngn.events.NgnStringEventArgs</a>	38

## Class Documentation

### org.doubango.ngn.sip.NgnSipSession.ConnectionState Enum Reference

---

#### Detailed Description

The connection state

---

The documentation for this enum was generated from the following file:

72 src/org/doubango/ngn/sip/NgnSipSession.java

### org.doubango.ngn.sip.NgnAVSession Class Reference

Inheritance diagram for org.doubango.ngn.sip.NgnAVSession:



## Public Member Functions

```
73 boolean makeCall (String remoteUri)
74 boolean makeVideoSharingCall (String remoteUri)
75 Context getContext ()
76 void setContext (Context context)
77 final View startVideoConsumerPreview ()
78 final View startVideoProducerPreview ()
79 boolean isSendingVideo ()
80 void toggleCamera ()
81 void setRotation (int rot)
82 void setSpeakerphoneOn (boolean speakerOn)
83 void toggleSpeakerphone ()
84 void setState (InviteState state)
85 boolean acceptCall ()
86 boolean hangUpCall ()
87 boolean holdCall ()
88 boolean resumeCall ()
89 boolean isLocalHeld ()
90 boolean isRemoteHeld ()
91 boolean sendDTMF (int digit)
```

## Static Public Member Functions

```
92 static NgnAVSession createOutgoingSession (NgnSipStack sipStack, NgnMediaType mediaType)
93 static NgnAVSession getSession (long id)
94 static int getSize ()
95 static boolean hasSession (long id)
96 static boolean hasActiveSession ()
97 static NgnAVSession getFirstActiveCallAndNot (long id)
98 static boolean makeAudioCall (String remoteUri, NgnSipStack sipStack)
99 static boolean makeAudioVideoCall (String remoteUri, NgnSipStack sipStack)
```

---

## Detailed Description

Audio/Video call session

---

## Member Function Documentation

**boolean [org.doubango.ngn.sip.NgnAVSession.acceptCall](#) ()**

Accepts an incoming audio/video call

**Returns:**

true is succeed and false otherwise

**See also:**

[hangUpCall\(\)](#)

**static [NgnAVSession org.doubango.ngn.sip.NgnAVSession.createOutgoingSession](#) ([NgnSipStack](#) sipStack, NgnMediaType mediaType) [static]**

Creates an outgoing audio/video call session.

**Parameters:**

<i>sipStack</i>	the IMS/SIP stack to use to make the call
<i>mediaType</i>	the media type.

**Returns:**

an audio/video session

**See also:**

[makeAudioCall\(\)](#) [makeAudioVideoCall\(\)](#)

**Context** [org.doubango.ngn.sip.NgnAVSession.getContext \(\)](#)

Gets the context associated to this session. Only used for video session to track the SurfaceView lifecycle

**Returns:**

the context

**static** [NgnAVSession org.doubango.ngn.sip.NgnAVSession.getFirstActiveCallAndNot](#) (long *id*) [**static**]

Gets the first active audio/video session with an id different than the one specified as parameter

**Parameters:**

<i>id</i>	the id of the session to exclude from the search
-----------	--

**Returns:**

an audio/video session matching the criteria or null if no one exist

**static** [NgnAVSession org.doubango.ngn.sip.NgnAVSession.getSession](#) (long *id*) [**static**]

Retrieves an audio/video session by id.

**Parameters:**

<i>id</i>	the id of the audio/video session to retrieve
-----------	---

**Returns:**

an audio/video session with the specified id if exist and null otherwise

**static int** [org.doubango.ngn.sip.NgnAVSession.getSize](#) () [**static**]

Gets the number of pending audio/video sessions. These sessions could be active or not.

**Returns:**

the number of pending audio/video sessions.

**See also:**

[hasActiveSession\(\)](#)

**boolean** [org.doubango.ngn.sip.NgnAVSession.hangUpCall](#) ()

Ends an audio/video call. The call could be in any state: incoming, outgoing, incall, ...

**Returns:**

true if succeed and false otherwise

**static boolean** [org.doubango.ngn.sip.NgnAVSession.hasActiveSession](#) () [**static**]

Check whether we have at least one active audio/video session.

**Returns:**

true if exist and false otherwise

**static boolean** [org.doubango.ngn.sip.NgnAVSession.hasSession](#) (long *id*) [**static**]

Checks whether we already have an audio/video session with the specified id.

**Parameters:**

<i>id</i>	the id of the session to look for
-----------	-----------------------------------

**Returns:**

true if exist and false otherwise

**boolean** [org.doubango.ngn.sip.NgnAVSession.holdCall](#) ()

Puts the call on hold. At any time you can check if the call is held or not by using [isLocalHeld\(\)](#)

**Returns:**

true if succeed and false otherwise

**See also:**

[resumeCall\(\)](#) [isLocalHeld\(\)](#) [isRemoteHeld\(\)](#) [resumeCall\(\)](#)

**boolean** [org.doubango.ngn.sip.NgnAVSession.isLocalHeld](#) ()

Checks whether the call is locally held or not. You should use [resumeCall\(\)](#) to resume the call.

**Returns:**

true if locally held and false otherwise

**See also:**

[isRemoteHeld\(\)](#)

Reimplemented from [org.doubango.ngn.sip.NgnInviteSession](#).

**boolean** [org.doubango.ngn.sip.NgnAVSession.isRemoteHeld](#) ()

Checks whether the call is remotely held or not

**Returns:**

true if the call is remotely held and false otherwise

**See also:**

[isLocalHeld\(\)](#)

Reimplemented from [org.doubango.ngn.sip.NgnInviteSession](#).

**boolean** [org.doubango.ngn.sip.NgnAVSession.isSendingVideo](#) ()

Checks whether we are sending video or not

**Returns:**

true if we are already sending video and false otherwise

**static boolean** [org.doubango.ngn.sip.NgnAVSession.makeAudioCall](#) (String *remoteUri*, [NgnSipStack sipStack](#)) [**static**]

Places an audio call. Even if the NGN engine supports multi-line calls it's recommended to check that there is no active call before trying to make new one. You can use [hasActiveSession\(\)](#) function to check there is already an active audio/video session. Putting

the current active call in hold before placing the new one could also be a recommended solution.

**Parameters:**

<i>remoteUri</i>	the remote party uri. Could be a SIP/TEL uri, nomadic number, MSISDN number, ... example: sip: <a href="mailto:test@doubango.org">test@doubango.org</a> , tel:+33600000000, 78888667, ...
<i>sipStack</i>	the SIP/IMS stack to use

**Returns:**

true if the call has been successfully placed and false otherwise

**See also:**

[createOutgoingSession\(\)](#) [makeAudioVideoCall\(\)](#)

**static boolean [org.doubango.ngn.sip.NgnAVSession.makeAudioVideoCall](#) (String *remoteUri*, [NgnSipStack](#) *sipStack*) [static]**

Places an audio/video call. Even if the NGN engine supports multi-line calls it's recommended to check that there is no active call before trying to make new one. You can use [hasActiveSession\(\)](#) function to check there is already an active audio/video session. Putting the current active call in hold before placing the new one could also be a recommended solution.

**Parameters:**

<i>remoteUri</i>	the remote party uri. Could be a SIP/TEL uri, nomadic number, MSISDN number, ... example: sip: <a href="mailto:test@doubango.org">test@doubango.org</a> , tel:+33600000000, 78888667, ...
<i>sipStack</i>	the SIP/IMS stack to use

**Returns:**

true if the call has been successfully placed and false otherwise

**See also:**

[createOutgoingSession\(\)](#) [makeAudioCall\(\)](#)

**boolean [org.doubango.ngn.sip.NgnAVSession.makeCall](#) (String *remoteUri*)**

Makes an audio/video call. The call type depends on the mediaType define in the session object.

**Parameters:**

<i>remoteUri</i>	the remote party uri. Could be a SIP/TEL uri, nomadic number, MSISDN number, ... example: sip: <a href="mailto:test@doubango.org">test@doubango.org</a> , tel:+33600000000, 78888667, ...
------------------	---

**Returns:**

true if the call succeed and false otherwise

**See also:**

[createOutgoingSession\(\)](#) [makeAudioCall\(\)](#) [makeAudioVideoCall\(\)](#)

**boolean [org.doubango.ngn.sip.NgnAVSession.makeVideoSharingCall](#) (String *remoteUri*)**

Starts video sharing session

**Parameters:**

<i>remoteUri</i>	the remote party uri. Could be a SIP/TEL uri, nomadic number, MSISDN number, ... example: sip: <a href="mailto:test@doubango.org">test@doubango.org</a> , tel:+33600000000, 78888667, ...
------------------	---

**Returns:**

true if the call succeed and false otherwise

**boolean** [org.doubango.ngn.sip.NgnAVSession.resumeCall](#) ()

Resumes a call. The call should be previously held using [holdCall\(\)](#)

**Returns:**

true is succeed and false otherwise

**See also:**

[holdCall\(\)](#) [isLocalHeld\(\)](#) [isRemoteHeld\(\)](#)

**boolean** [org.doubango.ngn.sip.NgnAVSession.sendDTMF](#) (int *digit*)

Sends DTMF digit. The session must be active (incoming, outgoing, incall, ...) in order to try to send DTMF digits.

**Parameters:**

<i>digit</i>	the digit to send
--------------	-------------------

**Returns:**

true if succeed and false otherwise

**void** [org.doubango.ngn.sip.NgnAVSession.setContext](#) (Context *context*)

Sets a context to associated to this session

**Parameters:**

<i>context</i>	the context
----------------	-------------

**void** [org.doubango.ngn.sip.NgnAVSession.setRotation](#) (int *rot*)

Sets the local video rotation angle

**Parameters:**

<i>rot</i>	rotation angle in degree
------------	--------------------------

**void** [org.doubango.ngn.sip.NgnAVSession.setSpeakerphoneOn](#) (boolean *speakerOn*)

Enables or disables the speakerphone

**Parameters:**

<i>speakerOn</i>	true to enable the speakerphone and false to disable it
------------------	---

**void** [org.doubango.ngn.sip.NgnAVSession.setState](#) (InviteState *state*)

Sets the session state

**Parameters:**

<i>state</i>	the new session state
--------------	-----------------------

Reimplemented from [org.doubango.ngn.sip.NgnInviteSession](#).

**final View** [org.doubango.ngn.sip.NgnAVSession.startVideoConsumerPreview](#) ()

Starts the video consumer. A video consumer view used to display the video stream sent from the remote party. It's up to you to embed this view into a layout (LinearLayout, RelativeLayout, FrameLayout, ...) in order to display it.

**Returns:**

the view where the remote video stream will be displayed

**final View** [org.doubango.ngn.sip.NgnAVSession.startVideoProducerPreview \(\)](#)

Starts the video producer. A video producer is any device capable to generate video frames. It's likely a video camera (front facing or rear). The view associated to the producer is used as a feedback to show the local video stream sent to the remote party. It's up to you to embed this view into a layout (LinearLayout, RelativeLayout, FrameLayout, ...) in order to display it.

**Returns:**

the view where the local video stream will be displayed

**void** [org.doubango.ngn.sip.NgnAVSession.toggleCamera \(\)](#)

Switch from rear to front-facing camera or vice-versa

**void** [org.doubango.ngn.sip.NgnAVSession.toggleSpeakerphone \(\)](#)

Toggles the speakerphone. Enable it if disabled and vice-versa

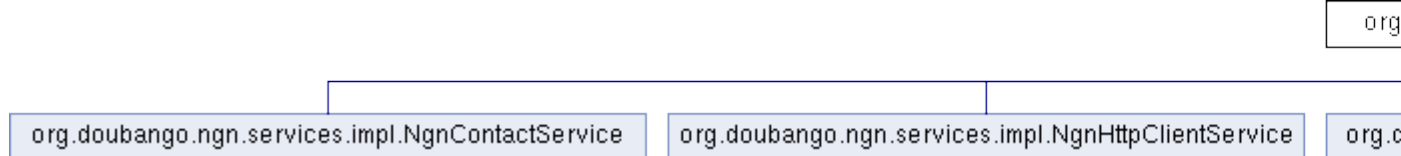
---

**The documentation for this class was generated from the following file:**

100 src/org/doubango/ngn/sip/NgnAVSession.java

## org.doubango.ngn.services.impl.NgnBaseService Class Reference

Inheritance diagram for org.doubango.ngn.services.impl.NgnBaseService:



---

### Detailed Description

Base class for all services

---

The documentation for this class was generated from the following file:

101 src/org/doubango/ngn/services/impl/NgnBaseService.java

## org.doubango.ngn.model.NgnContact Class Reference

Inherits org.doubango.ngn.utils.NgnObservableObject.

### Public Member Functions

102 [NgnContact](#) (int id, String displayName)  
103 int [getId](#) ()  
104 List< NgnPhoneNumber > [getPhoneNumbers](#) ()  
105 List< NgnEmail > [getEmails](#) ()  
106 String [getPrimaryNumber](#) ()

107 void [addPhoneNumber](#) (PhoneType type, String number, String description)  
108 void [addEmail](#) (NgnEmail.EmailType type, String value, String description)  
109 void [setDisplayDisplayName](#) (String displayName)  
110 String [getDisplayDisplayName](#) ()  
111 Bitmap [getPhoto](#) ()

---

## Detailed Description

Contact class defining an entity from the native address book or XCAP server.

---

## Constructor & Destructor Documentation

[org.doubango.ngn.model.NgnContact.NgnContact](#) (int *id*, String *displayName*)

Creates new address book

### Parameters:

<i>id</i>	a unique id defining this contact
<i>displayName</i>	the contact's display name

---

## Member Function Documentation

void [org.doubango.ngn.model.NgnContact.addEmail](#) (NgnEmail.EmailType *type*, String *value*, String *description*)

Attach a new email to this contact

### Parameters:

<i>type</i>	the type of the email to add
<i>value</i>	the actual value of the email
<i>description</i>	a short description

void [org.doubango.ngn.model.NgnContact.addPhoneNumber](#) (PhoneType *type*, String *number*, String *description*)

Attach a new phone number to this contact

### Parameters:

<i>type</i>	the type of the phone number to add
<i>number</i>	the actual value of the phone number
<i>description</i>	a short description

String [org.doubango.ngn.model.NgnContact.getDisplayName](#) ()

Gets the contact's display name

### Returns:

the contact's display name

List<NgnEmail> [org.doubango.ngn.model.NgnContact.getEmails](#) ()

Gets all emails associated to this contact

**Returns:**

list of all emails associated to this contact

**int** [org.doubango.ngn.model.NgnContact.getId \(\)](#)

Gets the id of the contact

**Returns:**

a unique id defining this contact

**List<NgnPhoneNumber>** [org.doubango.ngn.model.NgnContact.getPhoneNumbers \(\)](#)

Gets all phone numbers associated to this contact

**Returns:**

list of all numbers associated to this contact

**Bitmap** [org.doubango.ngn.model.NgnContact.getPhoto \(\)](#)

Gets the photo associated to this contact

**Returns:**

a bitmap representing the contact's photo

**String** [org.doubango.ngn.model.NgnContact.getPrimaryNumber \(\)](#)

Gets the default/primary phone number value. Most likely the mobile number

**Returns:**

the contact's primary number

**void** [org.doubango.ngn.model.NgnContact.setDisplayName \(String displayName\)](#)

Sets the contact's display name value

**Parameters:**

<i>displayName</i>	the new display name to assign to the contact
--------------------	---

---

**The documentation for this class was generated from the following file:**

112 src/org/doubango/ngn/model/NgnContact.java

## **org.doubango.ngn.services.impl.NgnContactService Class Reference**

Inheritance diagram for org.doubango.ngn.services.impl.NgnContactService:

---

### **Detailed Description**

Service used to retrieve contacts from the native address book.

---

**The documentation for this class was generated from the following file:**

113 src/org/doubango/ngn/services/impl/NgnContactService.java



## org.doubango.ngn.NgnEngine Class Reference

---

### Detailed Description

Next Generation Network Engine. This is the main entry point to have access to all services (SIP, XCAP, MSRP, History, ...). Anywhere in the code you can get an instance of the engine by calling getInstance() function.

---

The documentation for this class was generated from the following file:

114 src/org/doubango/ngn/NgnEngine.java

## org.doubango.ngn.events.NgnEventArgs Class Reference

Inheritance diagram for org.doubango.ngn.events.NgnEventArgs:

---

### Detailed Description

Base class for all events

---

The documentation for this class was generated from the following file:

115 src/org/doubango/ngn/events/NgnEventArgs.java

## org.doubango.ngn.services.impl.NgnHttpClientService Class Reference

Inheritance diagram for org.doubango.ngn.services.impl.NgnHttpClientService:

---

### Detailed Description

HTTP/HTTPS service.

---

The documentation for this class was generated from the following file:

116 src/org/doubango/ngn/services/impl/NgnHttpClientService.java

## org.doubango.ngn.events.NgnInviteEventArgs Class Reference

Inheritance diagram for org.doubango.ngn.events.NgnInviteEventArgs:

---

### Detailed Description

Event argument for SIP INVITE sessions

---

The documentation for this class was generated from the following file:

117 src/org/doubango/ngn/events/NgnInviteEventArgs.java

## org.doubango.ngn.events.NgnInviteEventTypes Enum Reference

---

### Detailed Description

List of all supported types associated to SIP INVITE event arguments

---

The documentation for this enum was generated from the following file:

118 src/org/doubango/ngn/events/NgnInviteEventTypes.java

## org.doubango.ngn.sip.NgnInviteSession Class Reference

Inheritance diagram for org.doubango.ngn.sip.NgnInviteSession:

### Public Member Functions

119 [NgnInviteSession](#) ([NgnSipStack](#) sipStack)  
120 NgnMediaType [getMediaType](#) ()  
121 InviteState [getState](#) ()  
122 void [setState](#) (InviteState state)  
123 boolean [isActive](#) ()  
124 MediaSessionMgr [getMediaSessionMgr](#) ()

---

### Detailed Description

Generic INVITE session. Could be either audio/video or MSRP session. This is an abstract class and you should only use it if you want to define your own session.

---

### Constructor & Destructor Documentation

[org.doubango.ngn.sip.NgnInviteSession.NgnInviteSession](#) ([NgnSipStack](#) sipStack)

Creates new Invite session

#### Parameters:

<i>sipStack</i>	the stack to use
-----------------	------------------

---

### Member Function Documentation

MediaSessionMgr [org.doubango.ngn.sip.NgnInviteSession.getMediaSessionMgr](#) ()

Gets the media session manager associated to this session

#### Returns:

the media session manager

**NgnMediaType** [org.doubango.ngn.sip.NgnInviteSession.getMediaType \(\)](#)

Gets the media type

**Returns:**

the media type

**InviteState** [org.doubango.ngn.sip.NgnInviteSession.getState \(\)](#)

Gets the session state

**Returns:**

the session state

**boolean** [org.doubango.ngn.sip.NgnInviteSession.isActive \(\)](#)

Checks whether the session is active or not

**Returns:**

**void** [org.doubango.ngn.sip.NgnInviteSession.setState \(InviteState state\)](#)

Sets the session state

**Parameters:**

<i>state</i>	the new session state
--------------	-----------------------

Reimplemented in [org.doubango.ngn.sip.NgnAVSession](#).

---

The documentation for this class was generated from the following file:

125 src/org/doubango/ngn/sip/NgnInviteSession.java

## org.doubango.ngn.sip.NgnMessagingSession Class Reference

Inheritance diagram for org.doubango.ngn.sip.NgnMessagingSession:

### Public Member Functions

126 boolean [SendBinaryMessage](#) (String text, String SMSC)

127 boolean [sendTextMessage](#) (String text, String contentType)

128 boolean [accept](#) ()

129 boolean [reject](#) ()

---

### Detailed Description

Messaging session used to send Pager Mode IM (SIP MESSAGE)

---

### Member Function Documentation

**boolean** [org.doubango.ngn.sip.NgnMessagingSession.accept \(\)](#)

Accepts the message (sends 200 OK).

**Returns:**

true if succeed and false otherwise

**boolean** [org.doubango.ngn.sip.NgnMessagingSession.reject \(\)](#)

Reject the message (sends 603 Decline)

**Returns:**

true if succeed and false otherwise

**boolean** [org.doubango.ngn.sip.NgnMessagingSession.SendBinaryMessage](#) (String *text*, String *SMSC*)

Sends binary SMS (3gpp) using SIP MESSAGE request

**Parameters:**

<i>text</i>	the text (utf-8) to send.
<i>SMSC</i>	the address (PSI) of the SMS center

**Returns:**

true if succeed and false otherwise

**See also:**

[sendTextMessage\(\)](#)

**boolean** [org.doubango.ngn.sip.NgnMessagingSession.sendTextMessage](#) (String *text*, String *contentType*)

Send plain text message using SIP MESSAGE request

**Parameters:**

<i>text</i>	
-------------	--

**Returns:**

true if succeed and false otherwise

**See also:**

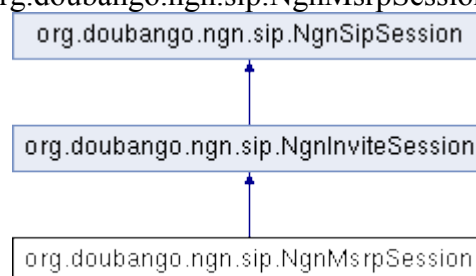
[SendBinaryMessage\(\)](#)

The documentation for this class was generated from the following file:

130 src/org/doubango/ngn/sip/NgnMessagingSession.java

## org.doubango.ngn.sip.NgnMsrpSession Class Reference

Inheritance diagram for org.doubango.ngn.sip.NgnMsrpSession:



## Detailed Description

MSRP session used to share content or send large IM

---

The documentation for this class was generated from the following file:

131 src/org/doubango/ngn/sip/NgnMrpSession.java

## org.doubango.ngn.NgnNativeService Class Reference

Inherits Service.

---

## Detailed Description

Android native service running in the background. This service is started but the engine.

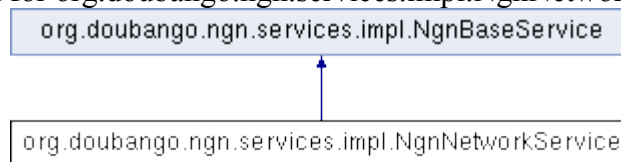
---

The documentation for this class was generated from the following file:

132 src/org/doubango/ngn/NgnNativeService.java

## org.doubango.ngn.services.impl.NgnNetworkService Class Reference

Inheritance diagram for org.doubango.ngn.services.impl.NgnNetworkService:



---

## Detailed Description

Network service.

---

The documentation for this class was generated from the following file:

133 src/org/doubango/ngn/services/impl/NgnNetworkService.java

## org.doubango.ngn.sip.NgnPresenceStatus Enum Reference

---

## Detailed Description

Presence status

---

The documentation for this enum was generated from the following file:

134 src/org/doubango/ngn/sip/NgnPresenceStatus.java

## org.doubango.ngn.media.NgnProxyAudioConsumer Class Reference

Inheritance diagram for org.doubango.ngn.media.NgnProxyAudioConsumer:

---

### Detailed Description

MyProxyAudioConsumer

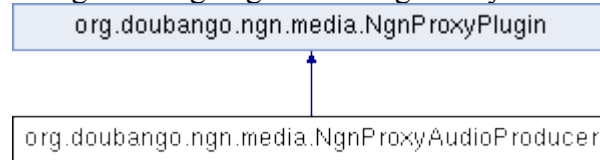
---

The documentation for this class was generated from the following file:

135 src/org/doubango/ngn/media/NgnProxyAudioConsumer.java

## org.doubango.ngn.media.NgnProxyAudioProducer Class Reference

Inheritance diagram for org.doubango.ngn.media.NgnProxyAudioProducer:



---

### Detailed Description

MyProxyAudioProducer

---

The documentation for this class was generated from the following file:

136 src/org/doubango/ngn/media/NgnProxyAudioProducer.java

## org.doubango.ngn.media.NgnProxyPlugin Class Reference

Inheritance diagram for org.doubango.ngn.media.NgnProxyPlugin:

---

### Detailed Description

MyProxyPlugin

---

The documentation for this class was generated from the following file:

137 src/org/doubango/ngn/media/NgnProxyPlugin.java

## org.doubango.ngn.media.NgnProxyVideoConsumerGL Class Reference

Inherits org.doubango.ngn.media.NgnProxyVideoConsumer.

---

## Detailed Description

Video consumer using OpenGL ES 2.0

---

The documentation for this class was generated from the following file:

138 src/org/doubango/ngn/media/NgnProxyVideoConsumerGL.java

## org.doubango.ngn.media.NgnProxyVideoConsumerSV Class Reference

Inherits org.doubango.ngn.media.NgnProxyVideoConsumer.

---

## Detailed Description

Video consumer using SurfaceView

---

The documentation for this class was generated from the following file:

139 src/org/doubango/ngn/media/NgnProxyVideoConsumerSV.java

## org.doubango.ngn.media.NgnProxyVideoProducer Class Reference

Inheritance diagram for org.doubango.ngn.media.NgnProxyVideoProducer:

---

## Detailed Description

MyProxyVideoProducer

---

The documentation for this class was generated from the following file:

140 src/org/doubango/ngn/media/NgnProxyVideoProducer.java

## org.doubango.ngn.sip.NgnRegistrationSession Class Reference

Inheritance diagram for org.doubango.ngn.sip.NgnRegistrationSession:

## Public Member Functions

141 [NgnRegistrationSession](#) ([NgnSipStack](#) sipStack)

142 boolean [register](#) ()

143 boolean [unregister](#) ()

---

## Detailed Description

Registration state

---

## Constructor & Destructor Documentation

[org.doubango.ngn.sip.NgnRegistrationSession.NgnRegistrationSession](#) ([NgnSipStack sipStack](#))

Creates new registration session

### Parameters:

<a href="#">sipStack</a>	the stack to use to create the session
--------------------------	--

---

## Member Function Documentation

boolean [org.doubango.ngn.sip.NgnRegistrationSession.register](#) ()

Sends SIP REGISTER request

### Returns:

true if succeed and false otherwise

boolean [org.doubango.ngn.sip.NgnRegistrationSession.unregister](#) ()

Unregisters (SIP REGISTER with expires=0)

### Returns:

true if succeed and false otherwise

---

The documentation for this class was generated from the following file:

144 src/org/doubango/ngn/sip/NgnRegistrationSession.java

## org.doubango.ngn.sip.NgnSipSession Class Reference

Inheritance diagram for org.doubango.ngn.sip.NgnSipSession:

### Classes

145 enum [ConnectionState](#)

### Public Member Functions

146 int [incRef](#) ()

147 int [decRef](#) ()

148 long [getId](#) ()

149 [NgnSipStack](#) [getStack](#) ()

150 boolean [addHeader](#) (String name, String value)

151 boolean [removeHeader](#) (String name)

152 boolean [addCaps](#) (String name)

153 boolean [addCaps](#) (String name, String value)



154 boolean [removeCaps](#) (String name)  
 155 boolean [isConnected](#) ()  
 156 void [setConnectionState](#) ([ConnectionState](#) state)  
 157 [ConnectionState](#) [getConnectionState](#) ()  
 158 String [getFromUri](#) ()  
 159 boolean [setFromUri](#) (String uri)

## Protected Member Functions

160 [NgnSipSession](#) ([NgnSipStack](#) sipStack)

---

## Detailed Description

Abstract class defining a SIP Session (Registration, Subscription, Publication, Call, ...)

---

## Constructor & Destructor Documentation

[org.doubango.ngn.sip.NgnSipSession.NgnSipSession](#) ([NgnSipStack](#) sipStack)  
 [protected]

Creates new SIP session

### Parameters:

<i>sipStack</i>	the sip stack to use to create the session
-----------------	--

---

## Member Function Documentation

boolean [org.doubango.ngn.sip.NgnSipSession.addCaps](#) (String name)

Adds sip capabilities to the session. The capability will be added in a separate "Accept-Contact" header if the session is dialogless or in the "Contact" header otherwise

### Parameters:

<i>name</i>	the name of capability to add
-------------	-------------------------------

### Returns:

true if succeed and false otherwise

### See also:

[removeCaps\(\)](#)  
 mSipSession.addCaps ("+g.3gpp.smsip");

boolean [org.doubango.ngn.sip.NgnSipSession.addCaps](#) (String name, String value)

Adds sip capabilities to the session. The capability will be added in a separate "Accept-Contact" header if the session is dialogless or in the "Contact" header otherwise

### Parameters:

<i>name</i>	the name of capability to add
<i>value</i>	the value of the capability

### Returns:

true if succeed and false otherwise

**See also:**

[removeCaps\(\)](#)

```
mSipSession.addCaps("+g.3gpp.icsi-ref", "\"urn%3Aurn-7%3A3gpp-service.ims.icsi.mmtel\"");
```

**boolean** [org.doubango.ngn.sip.NgnSipSession.addHeader](#) (String *name*, String *value*)

Adds a new SIP header to the session

**Parameters:**

<i>name</i>	the name of the header
<i>value</i>	the value of the header

**Returns:**

true if succeed and false otherwise

**See also:**

[removeHeader\(\)](#)

```
mSipSession.addHeader("User-Agent", "IM-OMAv1.0");
```

**int** [org.doubango.ngn.sip.NgnSipSession.decRef](#) ()

Decrements the reference counting

**Returns:**

the new reference counting value

**See also:**

[incRef\(\)](#)

**ConnectionState** [org.doubango.ngn.sip.NgnSipSession.getConnectionState](#) ()

Gets the connection state of the session

**Returns:**

the connection state

**See also:**

[isConnected\(\)](#)

**String** [org.doubango.ngn.sip.NgnSipSession.getFromUri](#) ()

Gets the sip from uri

**Returns:**

the sip from uri

**long** [org.doubango.ngn.sip.NgnSipSession.getId](#) ()

Gets a unique identifier defining a session

**Returns:**

a unique identifier defining the session

**NgnSipStack** [org.doubango.ngn.sip.NgnSipSession.getStack](#) ()

Gets the associated SIP stack

**Returns:**

a SIP stack

**int** [org.doubango.ngn.sip.NgnSipSession.incRef](#) ()

Increments the reference counting

**Returns:**

the new reference counting value

**See also:**

[decRef\(\)](#)

**boolean** [org.doubango.ngn.sip.NgnSipSession.isConnected](#) ()

Checks whether the session established or not. For example, you can only send files when the session is connected. You can use [getConnectionState\(\)](#) to have the exact state

**Returns:**

true is session is established and false otherwise

**See also:**

[getConnectionState\(\)](#)

**boolean** [org.doubango.ngn.sip.NgnSipSession.removeCaps](#) (String *name*)

Removes a sip capability from the session

**Parameters:**

<i>name</i>	the name of the capability to remove
-------------	--------------------------------------

**Returns:**

true if succeed and false otherwise

**See also:**

[addCaps\(\)](#)

```
mSipSession.removeCaps("+g.3gpp.smsip");
```

**boolean** [org.doubango.ngn.sip.NgnSipSession.removeHeader](#) (String *name*)

Removes a SIP header from the session

**Parameters:**

<i>name</i>	the name of the sip header to remove
-------------	--------------------------------------

**Returns:**

true if succeed and false otherwise

**See also:**

[addHeader\(\)](#)

```
mSipSession.removeHeader("User-Agent");
```

**void** [org.doubango.ngn.sip.NgnSipSession.setConnectionState](#) ([ConnectionState](#) *state*)

Sets the connection state of the session. You should not call this function by yourself

**Parameters:**

<i>state</i>	the new state
--------------	---------------

**boolean** [org.doubango.ngn.sip.NgnSipSession.setFromUri](#) (String *uri*)

Sets the sip from uri

**Parameters:**

<i>uri</i>	the new sip from uri
------------	----------------------

**Returns:**

true if succeed and false otherwise

**See also:**

ref setToUri()

---

The documentation for this class was generated from the following file:

161 src/org/doubango/ngn/sip/NgnSipSession.java

## org.doubango.ngn.sip.NgnSipStack Class Reference

Inherits SipStack.

### Public Member Functions

162 [NgnSipStack](#) (SipCallback callback, String realmUri, String impiUri, String impuUri)

---

### Detailed Description

SIP/IMS Stack

---

### Constructor & Destructor Documentation

[org.doubango.ngn.sip.NgnSipStack.NgnSipStack](#) (SipCallback *callback*, String *realmUri*, String *impiUri*, String *impuUri*)

Creates new SIP/IMS Stack. You should use

**Parameters:**

<i>callback</i>	
<i>realmUri</i>	
<i>impiUri</i>	
<i>impuUri</i>	

---

The documentation for this class was generated from the following file:

163 src/org/doubango/ngn/sip/NgnSipStack.java

## org.doubango.ngn.services.impl.NgnSoundService Class Reference

Inheritance diagram for org.doubango.ngn.services.impl.NgnSoundService:

---

## Detailed Description

Sound service.

---

The documentation for this class was generated from the following file:

164 src/org/doubango/ngn/services/impl/NgnSoundService.java

## org.doubango.ngn.events.NgnStackEventArgs Class Reference

Inheritance diagram for org.doubango.ngn.events.NgnStackEventArgs:



---

## Detailed Description

Event argument associated to the stack

---

The documentation for this class was generated from the following file:

165 src/org/doubango/ngn/events/NgnStackEventArgs.java

## org.doubango.ngn.events.NgnStackEventTypes Enum Reference

---

## Detailed Description

List of all supported types associated to Stack event arguments

**See also:**

NgnStackEventArgs()

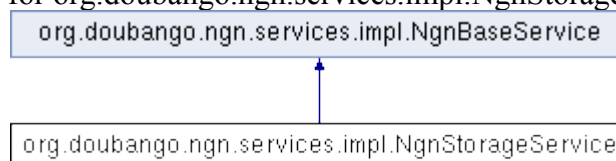
---

The documentation for this enum was generated from the following file:

166 src/org/doubango/ngn/events/NgnStackEventTypes.java

## org.doubango.ngn.services.impl.NgnStorageService Class Reference

Inheritance diagram for org.doubango.ngn.services.impl.NgnStorageService:



## Detailed Description

Storage service.

---

The documentation for this class was generated from the following file:

167 src/org/doubango/ngn/services/impl/NgnStorageService.java

## org.doubango.ngn.events.NgnStringEventArgs Class Reference



---

## Detailed Description

Generic event argument containing short string

---

The documentation for this class was generated from the following file:

168 src/org/doubango/ngn/events/NgnStringEventArgs.java

## Index

INDEX