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 iDoubs: 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 Enhanced Address Book (XCAP storage, authorizations, presence, ...)
10 GSMA Rich Communication Suite release 3
11 Partial supports for One Voice Profile V1.0.0 (GSMA VoLTE)
12 Partial supports for MMTel UNI (used by GSMA RCS and GSMA VoLTE)

13 IMS-AKA registration (both AKA-v1 and AKA-v2), Digest MD5, Basic
14 3GPP Early IMS Security (3GPP TS 33.978)
15 Proxy-CSCF discovery using DNS NAPTR+SRV
16 Private extension headers for 3GPP
17 Service Route discovery
18 Subscription to reg event package (Honoring network initiated (re/de/un)-registration events)

19 3GPP SMS Over IP (3GPP TS 23.038, 24.040, 24.011, 24.341 and 24.451)
20 Voice Call (G729AB1, AMR-NB, iLBC, GSM, PCMA, PCMU, Speex-NB)
21 Video Call (H264, MP4V-ES, Theora, H.263, H.263-1998, H.261)
22 DTMF (RFC 4733)
23 QoS negotiation using Preconditions (RFC 3312, 4032 and 5027)
24 SIP Session Timers (RFC 4028)
25 Provisional Response Acknowledgments (PRACK)
26 Communication Hold (3GPP TS 24.610)
27 Message Waiting Indication (3GPP TS 24.606)
28 Calling E.164 numbers by using ENUM protocol (RFC 3761)
29 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)

30 One2One and Group Chat
31 File Transfer and Content sharing

Architecture

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

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 scoop 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 the 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.

License

android-ngn-stack is a NGN (Next Generation Network) stack to develop IMS/RCS/VoLTE applications for Android devices. The stack is based on Doubango project.

Android Permissions

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

Required permissions:

```
        <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"/>
```

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();
```

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.equ
als(action)){
                                       NgnRegistrationEventArgs args = 
intent.getParcelableExtra(NgnEventArgs.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);
```

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 you 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*). 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 IPMU 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.NgnApplication....12
org.doubango.ngn.sip.NgnAVSession....13
org.doubango.ngn.services.impl.NgnBaseService....18
- org.doubango.ngn.services.impl.NgnContactService....21
- org.doubango.ngn.services.impl.NgnHttpClientService....24
- org.doubango.ngn.services.impl.NgnNetworkService....25
- org.doubango.ngn.services.impl.NgnSoundService....27
- org.doubango.ngn.services.impl.NgnStorageService....28

org.doubango.ngn.media.NgnCameraProducer....19
org.doubango.ngn.model.NgnContact....19
org.doubango.ngn.NgnEngine....21
org.doubango.ngn.events.NgnEventArgs....24
- org.doubango.ngn.events.NgnInviteEventArgs....24
- org.doubango.ngn.events.NgnStackEventArgs....27
- org.doubango.ngn.events.NgnStringEventArgs....28

org.doubango.ngn.NgnNativeService....25
org.doubango.ngn.media.NgnProxyPlugin....26
- org.doubango.ngn.media.NgnProxyAudioConsumer....25
- org.doubango.ngn.media.NgnProxyAudioProducer....25
- org.doubango.ngn.media.NgnProxyVideoConsumer....26
- org.doubango.ngn.media.NgnProxyVideoProducer....26

org.doubango.ngn.sip.NgnSipStack....26

Class Index

Class List

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

org.doubango.ngn.NgnApplication 12
org.doubango.ngn.sip.NgnAVSession 13
org.doubango.ngn.services.impl.NgnBaseService 18
org.doubango.ngn.media.NgnCameraProducer 19
org.doubango.ngn.model.NgnContact 19
org.doubango.ngn.services.impl.NgnContactService 21
org.doubango.ngn.NgnEngine 21
org.doubango.ngn.events.NgnEventArgs 24
org.doubango.ngn.services.impl.NgnHttpClientService 24

org.doubango.ngn.events.NgnInviteEventArgs 24
org.doubango.ngn.NgnNativeService 25
org.doubango.ngn.services.impl.NgnNetworkService 25
org.doubango.ngn.media.NgnProxyAudioConsumer 25
org.doubango.ngn.media.NgnProxyAudioProducer 25
org.doubango.ngn.media.NgnProxyPlugin 26
org.doubango.ngn.media.NgnProxyVideoConsumer 26
org.doubango.ngn.media.NgnProxyVideoProducer 26
org.doubango.ngn.sip.NgnSipStack 26
org.doubango.ngn.services.impl.NgnSoundService 27
org.doubango.ngn.events.NgnStackEventArgs 27
org.doubango.ngn.services.impl.NgnStorageService 28
org.doubango.ngn.events.NgnStringEventArgs 28

Class Documentation

org.doubango.ngn.NgnApplication Class Reference

Inherits android::app::Application.

Static Public Member Functions

32 static Context getContext ()
33 static int getSDKVersion ()
34 static boolean useSetModeToHackSpeaker ()
35 static boolean isSamsung ()
36 static boolean isHTC ()

Detailed Description

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

Member Function Documentation

static Context org.doubango.ngn.NgnApplication.getContext () `[static]`

Retrieve application's context

Returns:

Android context

static int org.doubango.ngn.NgnApplication.getSDKVersion () `[static]`

Gets Android SDK version

Returns:

Android SDK version used to build the project

static boolean org.doubango.ngn.NgnApplication.isHTC () `[static]`

Whether the stack is running on a HTC device

Returns:

true if the stack is running on a HTC device and false otherwise

static boolean org.doubango.ngn.NgnApplication.isSamsung () `[static]`

Whether the stack is running on a Samsung device

Returns:

true if the stack is running on a Samsung device and false otherwise

static boolean org.doubango.ngn.NgnApplication.useSetModeToHackSpeaker () `[static]`

Whether we need special hack for buggy speaker. For example, all Samsung devices need to be hacked.

Returns:

true if we need to apply the hack and false otherwise

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

37 src/org/doubango/ngn/NgnApplication.java

org.doubango.ngn.sip.NgnAVSession Class Reference

Inherits org::doubango::ngn::sip::NgnInviteSession.

Public Member Functions

38 boolean makeCall (String remoteUri)
39 boolean makeVideoSharingCall (String remoteUri)
40 Context getContext ()
41 void setContext (Context context)
42 final View startVideoConsumerPreview ()
43 final View startVideoProducerPreview ()
44 boolean isSendingVideo ()
45 void toggleCamera ()
46 void setRotation (int rot)
47 void setSpeakerphoneOn (boolean speakerOn)
48 void toggleSpeakerphone ()
49 boolean acceptCall ()
50 boolean hangUpCall ()
51 boolean holdCall ()
52 boolean resumeCall ()
53 boolean isLocalHeld ()
54 boolean isRemoteHeld ()
55 boolean sendDTMF (int digit)

Static Public Member Functions

56 static NgnAVSession createOutgoingSession (NgnSipStack sipStack, NgnMediaType mediaType)

57. static NgnAVSession getSession (long id)
58. static int getSize ()
59. static boolean hasSession (long id)
60. static boolean hasActiveSession ()
61. static NgnAVSession getFirstActiveCallAndNot (long id)
62. static boolean makeAudioCall (String remoteUri, NgnSipStack sipStack)
63. 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:

sipStack	the IMS/SIP stack to use to make the call
mediaType	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:

id	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:

id	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:

id	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 held or not. You should use resumeCall() to resume the call.

Returns:

true if locally held and false otherwise

See also:

isRemoteHeld()

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()

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

Parameters:

remoteUri	the remote party uri. Could be a SIP/TEL uri, nomadic number, MSISDN number, ... example: sip:test@doubango.org, tel:+33600000000, 78888667, ...
sipStack	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. Event 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 active call in hold before placing the new one could also be a recommended solution.

Parameters:

remoteUri	the remote party uri. Could be a SIP/TEL uri, nomadic number, MSISDN number, ... example: sip:test@doubango.org, tel:+33600000000, 78888667, ...
sipStack	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:

remoteUri	the remote party uri. Could be a SIP/TEL uri, nomadic number, MSISDN number, ... example: sip:test@doubango.org, 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:

remoteUri	the remote party uri. Could be a SIP/TEL uri, nomadic number, MSISDN number, ... example: sip:test@doubango.org, 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:

digit	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:

context	the context

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

Sets the local video rotation angle

Parameters:

rot	rotation angle in degree

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

Enables or disables the speakerphone

Parameters:

speakerOn	true to enable the speakerphone and false to disable it

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, RelativeLayou, 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, RelativeLayou, 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 font-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:

64 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:

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

org.doubango.ngn.media.NgnCameraProducer Class Reference

Detailed Description

MyCameraProducer

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

66 src/org/doubango/ngn/media/NgnCameraProducer.java

org.doubango.ngn.model.NgnContact Class Reference

Inherits org::doubango::ngn::utils::NgnObservableObject.

Public Member Functions

67 NgnContact (int id, String displayName)
68 int getId ()
69 List< NgnPhoneNumber > getPhoneNumbers ()
70 String getPrimaryNumber ()
71 void addPhoneNumber (PhoneType type, String number, String description)
72 void setDisplayName (String displayName)
73 String getDisplayName ()
74 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:

id	a unique id defining this contact
displayName	the contact's display name

Member Function Documentation

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

Attach a new phone number to this contact

Parameters:

type	the type of the phone number to add
number	the actual value of the phone number
description	a short description

String org.doubango.ngn.model.NgnContact.getDisplayName ()

Gets the contact's display name

Returns:

the contact's display name

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:

displayName	the new display name to assign to the contact

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

75 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:

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

org.doubango.ngn.NgnEngine Class Reference

Public Member Functions

77 synchronized boolean start ()
78 synchronized boolean stop ()
79 synchronized boolean isStarted ()
80 void setMainActivity (Activity mainActivity)
81 Activity getMainActivity ()
82 INgnConfigurationService getConfigurationService ()
83 INgnStorageService getStorageService ()
84 INgnNetworkService getNetworkService ()
85 INgnHttpClientService getHttpClientService ()
86 INgnContactService getContactService ()
87 INgnHistoryService getHistoryService ()
88 INgnSipService getSipService ()
89 INgnSoundService getSoundService ()
90 Class<?extends NgnNativeService > getNativeServiceClass ()

Static Public Member Functions

91 static NgnEngine getInstance ()

Protected Member Functions

92 NgnEngine ()

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.

Constructor & Destructor Documentation

org.doubango.ngn.NgnEngine.NgnEngine () `[protected]`

Default constructor for the NGN engine. You should never call this function from your code.

Instead you should use getInstance() .

See also:

getInstance()

Member Function Documentation

INgnConfigurationService org.doubango.ngn.NgnEngine.getConfigurationService ()

Gets the configuration service.

Returns:

the configuration service.

INgnContactService org.doubango.ngn.NgnEngine.getContactService ()

Gets the contact service

Returns:

the contact service

INgnHistoryService org.doubango.ngn.NgnEngine.getHistoryService ()

Gets the history service

Returns:

the history service

INgnHttpClientService org.doubango.ngn.NgnEngine.getHttpClientService ()

Gets the HTTP service

Returns:

the HTTP service

static NgnEngine org.doubango.ngn.NgnEngine.getInstance () `[static]`

Gets an instance of the NGN engine. You can call this function as many as you need and it will always return th same instance.

Returns:

An instance of the NGN engine.

Activity org.doubango.ngn.NgnEngine.getMainActivity ()

Gets the main activity.

Returns:

the main activity

See also:

setMainActivity()

Class<? extends NgnNativeService> org.doubango.ngn.NgnEngine.getNativeServiceClass ()

Gets the native service class

Returns:

the native service class

INgnNetworkService org.doubango.ngn.NgnEngine.getNetworkService ()

Gets the network service

Returns:

the network service

INgnSipService org.doubango.ngn.NgnEngine.getSipService ()

Gets the SIP service

Returns:

the sip service

INgnSoundService org.doubango.ngn.NgnEngine.getSoundService ()

Gets the sound service

Returns:

the sound service

INgnStorageService org.doubango.ngn.NgnEngine.getStorageService ()

Gets the storage service.

Returns:

the storage service.

synchronized boolean org.doubango.ngn.NgnEngine.isStarted ()

Checks whether the engine is started.

Returns:

true is the engine is running and false otherwise.

See also:

start() stop()

void org.doubango.ngn.NgnEngine.setMainActivity (Activity*mainActivity*)

Sets the main activity to use as context in order to query some native resources. It's up to you to call this function in order to retrieve the contacts for the ContactService.

Parameters:

mainActivity	The activity

See also:

getMainActivity()

synchronized boolean org.doubango.ngn.NgnEngine.start ()

Starts the engine. This function will start all underlying services (SIP, XCAP, MSRP, History, ...). You must call this function before trying to use any of the underlying services.

Returns:

true if all services have been successfully started and false otherwise

synchronized boolean org.doubango.ngn.NgnEngine.stop ()

Stops the engine. This function will stop all underlying services (SIP, XCAP, MSRP, History, ...).

Returns:

true if all services have been successfully stopped and false otherwise

---

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

93 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:

94 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:

95 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:

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

org.doubango.ngn.NgnNativeService Class Reference

Inherits android::app::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:

97 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:

98 src/org/doubango/ngn/services/impl/NgnNetworkService.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:

99 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:

100 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:

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

org.doubango.ngn.media.NgnProxyVideoConsumer Class Reference

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

Detailed Description

MyProxyVideoConsumer

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

102 src/org/doubango/ngn/media/NgnProxyVideoConsumer.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:

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

org.doubango.ngn.sip.NgnSipStack Class Reference

Inherits org::doubango::tinyWRAP::SipStack.

Public Member Functions

104 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:

callback	
realmUri	
impiUri	
impuUri	

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

105 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:

106 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:

107 src/org/doubango/ngn/events/NgnStackEventArgs.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:

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

org.doubango.ngn.events.NgnStringEventArgs Class Reference

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

Detailed Description

Generic event argument containing short string

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

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

Index

INDEX