Gigaset

Gigaset API Specification

Oct 2013

Ver0.2

Revision No.	Date	Page	Comments for revision
0.1	04/10/2013	-	First release
0.2	08/10/2013	2	Added table of contents with page numbers.

Contents

1 Contacts APIs

1.1Contacts API List

A contact API list is described in Figure 1-1.

Figure 1-1 Contact API List

package: com.maxwell.directoryhelper.contacts

Class	Method	Contents	Comments
ContactsHelper	getContactsHelperInstance	Returns the Static instance of the Contacts Helper which can be used for accessing all other APIs	
	getBasicContactsList	Returns all available contacts in the DB	
	getContact	Returns a single contact using the contact ID	
	getContact	Returns a single contact using the contact bean object	
	getBasicContactDetails	Returns a single contact using the phone number of the contact	
	getBasicContactDetails	Gives a call back after Look up for a contact uinsg phone number. Asynchronous API which is ideally can be used for LDAP, XML Lookup etc	
	getcompName	Returns the Company Name of a contact using the contactId	
	insertContact	Inserts a new Contacts into the Contacst DB	
	updateContact	Updates a contacts into the ContactsDB	
	deleteAContact	deletes a contacts from the DB	
	getPictureUri	Returns the Picture URI of the contacts if present	

	connectAndSearch	Connects to the server and searches a string in the LDAP server. Results are given in the call back public void searchEntryReturned(final SearchResultEntry searchEntry)	
	searchContact	Searche a contact using the string in the XML net directory, results are returned using the call back public void onContactLoaded(final ContactBean contact)	

1.2 ContactsHelper class

1.2.1 getContactsHelperInstance

Returns the Static instance of the Contacts Helper which can be used for accessing all other APIs.

- **Format**

```
public static ContactsHelper getContactsHelperInstance(
                Context context) throws LDAPException, Exception
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
context	Context	in	

- **Return**
  Returns the static instance of the contact helper.

- **Exceptions**
  LDAPException
  Exception

1.2.2 getBasicContactsList

Returns all available contacts in the DB

- **Format**

```
public List<ContactBean> getBasicContactsList()
                         throws DirectoryHelperException,
                         FileNotFoundException,
                         IOException, Exception
```

- **Parameters**
  None.

- **Return**
  Returns all available contacts in DB

- **Exceptions**
  DirectoryHelperException
  FileNotFoundException
  IOException

1.2.3 getContact

Returns a single contact using the contact ID

- **Format**

```
public ContactBean getContact(int contactId)
                    throws
                    DirectoryHelperException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
contactId	int	in	The contact ID of database.

- **Return**
  Returns a single contact using the contact ID

- **Exceptions**
  DirectoryHelperException

1.2.4 getContact

Returns a single contact using the contact bean object

- **Format**

```
public ContactBean getContact(
                        ContactBean rowContact)
                        throws
                        DirectoryHelperException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
rowContact	ContactBean	in	

- **Return**
  Returns a single contact using the contact bean object.

- **Exceptions**
  DirectoryHelperException

1.2.5 getBasicContactDetails

Returns a single contact using the phone number of the contact

- **Format**

```
public BasicContact getBasicContactDetails(
                        String phoneNumber)
                        throws
                        DirectoryHelperException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
phoneNumber	String	in	

- **Return**
  Returns a single contact using the phone number of the contact

- **Exceptions**
  DirectoryHelperException

1.2.6 getBasicContactDetails

Gives a call back after Look up for a contact using phone number. Asynchronous API which is ideally can be used for LDAP, XML Lookup etc

- **Format**

```
public void getBasicContactDetails(
                String phoneNumber,
                ContactListener contactListener)
                throws
                DirectoryHelperException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
phoneNumber	String	in	
contactListener	ContactListener	in	

- **Return**
  None.

- **Exceptions**
  DirectoryHelperException

1.2.7 getcompName

Returns the Company Name of a contact using the contactId.

- **Format**

```
public String getcompName(
                String contactId)
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
context	Context	in	

- **Return**
  Returns the Company Name of a contact using the contactId.

- **Exceptions**
  None.

1.2.8 insertContact

Inserts a new Contacts into the Contacst DB

- **Format**

```
public boolean insertContact(
                        ContactBean contactBean)
                        throws
                        DirectoryHelperException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
contactBean	ContactBean	in	

- **Return**
  Returns success or failure of contact insertion into DB.

- **Exceptions**
  DirectoryHelperException

1.2.9 updateContact

Updates a contacts into the ContactsDB.

- **Format**

```
public boolean updateContact(
                    ContactBean contactBean)
                    throws
                    DirectoryHelperException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
contactBean	ContactBean	in	

- **Return**
  Returns success or failure of contact DB update.

- **Exceptions**
  DirectoryHelperException

1.2.10updateContact

Deletes a contacts from the DB

- **Format**

```
public boolean deleteAContact(
                        int rawId)
                        throws
                        DirectoryHelperException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
rawId	int	in	

- **Return**
  Returns success or failure of deleting a contact from DB

- **Exceptions**
  DirectoryHelperException

1.2.11getPictureUri

Returns the Picture URI of the contacts if present.

- **Format**

```
public Uri getPictureUri(
                String phoneNumber)
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
phoneNumber	String	in	

- **Return**
  Returns the Picture URI of the contacts if present.

- **Exceptions**
  None.

1.2.12connectAndSearch

Connects to the server and searches a string in the LDAP server. Results are given in the call back public void searchEntryReturned(final SearchResultEntry searchEntry)

- **Format**

```
public AsyncRequestID connectAndSearch(
                                String text,
                                SearchResultListener context,
                                String type)
                                throws
                                LDAPException,
                                Exception
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
text	String	in	
context	SearchResultListener	in	
type	String		

- **Return**
  Returns asynchronous request ID.

- **Exceptions**
  LDAPException
  Exception

1.2.13searchContact

Search a contact using the string in the XML net directory, results are returned using the call back public void onContactLoaded(final ContactBean contact)

- **Format**

```
public void searchContact(
                String searchText,
                String directoryType)
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
searchText	String	in	
directoryType	String	in	

- **Return**
  None.

- **Exceptions**
  None.

2 Call Log APIs

2.1 Call Log API List

A Call Log API list is described in Figure 2-1.

Figure 2-1 Call Log API List

package: com.maxwell.directoryhelper.calllog

Class	Method	Contents	Comments
LocalCallList	getCallLogDetails	Returns the Call Logs in an synchronous way using the call back method public void onCallLogChanged(CallLogList callList);	
	deleteSingleCallLog	Deletes a single call log from DB, status is returned using the call back method public void onDeleteComplete(CallLogList result);	
	deleteMultipleCallLogs	Deletes a list of call logs from the DB, status is returned using the call back method public void onDeleteComplete(CallLogList result);	
	insertCallLogDetails	Insert a call log into DB	

2.2 LocalCallList class

2.2.1 getCallLogDetails

Returns the Call Logs in an synchronous way using the call back method public void onCallLogChanged(CallLogList callList);

- **Format**

```
public void getCallLogDetails()
```

- **Parameters**
  None.

- **Return**
  None.

- **Exceptions**
  None.

2.2.2 deleteSingleCallLog

Deletes a single call log from DB, status is returned using the call back method public void onDeleteComplete(CallLogList result);

- **Format**

```
public void deleteSingleCallLog(
                String callId)
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
callId	String	in	

- **Return**
  None.

- **Exceptions**
  None.

2.2.3 deleteMultipleCallLogs

Deletes a list of call logs from the DB, status is returned using the call back method public void onDeleteComplete(CallLogList result);

- **Format**

```
public void deleteMultipleCallLogs(
                List<String> callIds)
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
callIds	List<String>	in	

- **Return**
  None.

- **Exceptions**
  None.

2.2.4 insertCallLogDetails

Insert a call log into DB

- **Format**

```
public Uri insertCallLogDetails(
                BasicCallInfo callLog)
                throws
                DirectoryHelperException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
callLog	BasicCallInfo	in	

- **Return**
  Returns URI.

- **Exceptions**
  DirectoryHelperException

3 Phone Intent Specification

3.1 *android.intent.action.CALL*

Any application can used CALL Intent action to make a call via Maxwell's Phone app.

- **Example of use**

```
Intent callIntent = new Intent(Intent.ACTION_CALL);
callIntent.setData(Uri.parse("tel:" + phoneNum));
startActivity(callIntent);
```

- **Extra**
  None.

- **Data**
  Set a <Phone number to call> to the intent by setData method

4 Phone APIs

4.1 Phone API List

A Phone API list is described in Figure 4-1.

Figure 4-1 Phone API List

package: com.maxwell.phone.services

Class	Method	Contents	Comments
SipService	registerReceiver	Register a ResultReceiver for getting the callback.	
	unregisterReceiver	Unregister a ResultReceiver.	
	initSipStack	Used to initialize the doubango SIP stack. Registering audio and video producer consumer.	
	setConfig	Used to set the configuration in doubango stack. It will retrun success or failure code.	
	maxwellCreateTwoPartyCall	Can be used to make a two party call. It will return callID in the case of success otherwise failure error code will be return.	
	hangUp	Can be used to hangUp a call. It will retrun success or failure code.	
	holdCall	Can be used to hold a call. It will return success or failure code.	
	resumeCall	Can be used to resume a hold call. It will return success or failure code.	
	acceptCall	Can be used to accept a incoming call. It will return success or failure code.	

startCallTransfer	Can be used to intiate the Call transfer. It will return success or failure code.	
completeCallTransfer	Can be used to complete the call transfer. It will return success or failure code.	
cancelCallTransfer	Can be used to cancel a call transfer. It will return success or failure code.	
startBlindCallTransfer	Call be used to initiate a blind call transfer. It will return success or failure code.	
startVideo	Can be used to start a video for a call. It will return success or failure code.	
stopVideo	Can be used to stop a video for a call. It will return success or failure code.	
createConfernceFromTwoPartyCall	Can be used to create a conference call. It will return success or failure code.	
createConfernceFromUri	Can be used to create a conference call from a active call and Uri. It will return success or failure code.	
hangUpConference	Can be used to hangup a conference. It will return success or failure code.	
maxwellCreateTwoPartyVideoCall	Can be used to make a two party video call. It will return callID in the case of success otherwise failure error code will be return.	
doDivertConfig	Can be used to change the call divert settings.	

	addTwoPartyCallToConference	Can be used to add a active two party call into a active conference call. It will return callID in the case of success otherwise failure error code will be return.	
	addNumberToConference	Can be used to add a number into a active conference call. It will return callID in the case of success otherwise failure error code will be return.	
	holdConference	Can be used to hold a conference call. It will return success or failure code.	
	resumeConference	Can be used to resume a hold conference call. It will return success or failure code.	
	swapConference	Can be used to swap between an active and hold conference. It will return success or failure code.	

4.2 SipService class

4.2.1 registerReceiver

Register a ResultReceiver for getting the callback.

- **Format**

```
public void registerReceiver(
            String key,
            ResultReceiver resultReceiver)
            throws
            RemoteException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
key	String	in	
resultReceiver	ResultReceiver	in	

- **Return**
  None.

- **Exceptions**
  RemoteException

4.2.2 unregisterReceiver

Unregister a ResultReceiver.

- **Format**

```
public void unregisterReceiver(
                String className)
                throws
                RemoteException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
className	String	in	

- **Return**
  None.

- **Exceptions**
  RemoteException

4.2.3 initSipStack

Used to initialize the doubango SIP stack. Registering audio and video produce consumer.

- **Format**

```
public void initSipStack(
                final CopyDeviceDetails copyDeviceDetails,
                RequestType request)
                throws RemoteException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
copyDeviceDetails	CopyDeviceDetails	in	
request	RequestType	in	

- **Return**
  None.

- **Exceptions**
  RemoteException

4.2.4 setConfig

Used to set the configuration in doubango stack. It will return success or failure code.

- **Format**

```
public void setConfig(
                final RequestType request)
                throws RemoteException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
rowContact	ContactBean	in	

- **Return**
  None.

- **Exceptions**
  RemoteException

4.2.5 maxwellCreateTwoPartyCall

Can be used to make a two party call. It will return callID in the case of success otherwise failure error code will be return.

- **Format**

```
public void maxwellCreateTwoPartyCall(
                String calledURI,
                int accountId,
                RequestType request)
                throws RemoteException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
calledURI	String	in	
accountId	int	in	
request	RequestType	in	

- **Return**
  None.

- **Exceptions**
  RemoteException

4.2.6 hangUp

Can be used to hangUp a call. It will return success or failure code.

- **Format**

```
public void hangUp(
            int callId,
            RequestType request)
            throws RemoteException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
callId	int	in	
request	RequestType	in	

- **Return**
  None.

- **Exceptions**
  DirectoryHelperException

4.2.7 holdCall

Can be used to hold a call. It will return success or failure code.

- **Format**

```
public void holdCall(
                int callId,
                RequestType request)
                throws RemoteException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
callId	int	in	
request	RequestType	in	

- **Return**
  None.

- **Exceptions**
  RemoteException.

4.2.8 resumeCall

Can be used to resume a hold call. It will return success or failure code.

- **Format**

```
public void resumeCall(
                int callId,
                RequestType request)
                throws RemoteException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
callId	int	in	
request	RequestType	in	

- **Return**
  None.

- **Exceptions**
  RemoteException.

4.2.9 acceptCall

Can be used to accept a incoming call. It will return success or failure code.

- **Format**

```
public void acceptCall(
                int callId,
                RequestType request)
                throws RemoteException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
callId	int		
request	RequestType	in	

- **Return**
  None.

- **Exceptions**
  RemoteException.

4.2.10startCallTransfer

Can be used to intiate the Call transfer. It will return success or failure code.

- **Format**

```
public void startCallTransfer(
                final int accountId,
                final int callIDActiveCall,
                final String calledURI,
                final boolean isVideoFlag,
                RequestType request)
                throws RemoteException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
accountId	int	in	
callIDActive Call	int	in	
calledURI	String	in	
isVideoFlag	boolean	in	
RequestType	request	in	

- **Return**
  None.

- **Exceptions**
  RemoteException.

4.2.11completeCallTransfer

Can be used to complete the call transfer. It will return success or failure code.

.

- **Format**

```
public void completeCallTransfer(
                final int callIDActiveCall,
                final int callIDHoldCall,
                RequestType request)
                throws RemoteException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
callIDActiveCall	int	in	
callIDHoldCall	int	in	
request	RequestType	in	

- **Return**
  None

- **Exceptions**
  RemoteException.

4.2.12cancelCallTransfer

Can be used to cancel a call transfer. It will return success or failure code.

- **Format**

```
public void cancelCallTransfer(
                final int callIDActiveCall,
                final int callIDHoldCall,
                RequestType request)
                throws RemoteException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
callIDActiveCall	int	in	
callIDHoldCall	int	in	
request	RequestType	in	

- **Return**
  None.

- **Exceptions**
  RemoteException.

4.2.13startBlindCallTransfer

Call be used to initiate a blind call transfer. It will return success or failure code.

- **Format**

```
public void startBlindCallTransfer(
                final int sendReferCallId,
                final String calledUri,
                RequestType request)
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
sendReferCallId	int	in	
calledUri	String		
request	RequestType	in	

- **Return**
  None.

- **Exceptions**
  None.

4.2.14startVideo

Can be used to start a video for a call. It will return success or failure code.

- **Format**

```
public void startVideo(
                int callIDActiveCall,
                final RequestType request)
                throws RemoteException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
callIDActiveCall	int	in	
request	RequestType	in	

- **Return**
  None.

- **Exceptions**
  RemoteException.

4.2.15stopVideo

Can be used to stop a video for a call. It will return success or failure code.

- **Format**

```
public void stopVideo(
                int callIDActiveCall,
                final RequestType request)
                 throws RemoteException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
callIDActiveCall	int		
request	RequestType	in	

- **Return**
  None.

- **Exceptions**
  RemoteException.

4.2.16createConfernceFromTwoPartyCall

Can be used to create a conference call. It will return success or failure code.

- **Format**

```
public void createConfernceFromTwoPartyCall(
                final int activeCallId,
                final int holdCallId,
                 final RequestType request)
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
rowContact	ContactBean	in	

- **Return**
  None.

- **Exceptions**
  None.

4.2.17createConfernceFromUri

Can be used to create a conference call from a active call and Uri. It will return success or failure code.

- **Format**

```
public void createConfernceFromUri(
                final int activeCallId,
                final String number,
                final RequestType request)
                throws RemoteException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
activeCallId	int	in	
number	String	in	
request	RequestType	in	

- **Return**
  None.

- **Exceptions**
  RemoteException.

4.2.18hangUpConference

Can be used to hangup a conference. It will return success or failure code.

- **Format**

```
public void hangUpConference(
                int confId,
                RequestType request)
                throws RemoteException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
confId	int	in	
request	RequestType	in	

- **Return**
  None.

- **Exceptions**
  RemoteException.

4.2.19maxwellCreateTwoPartyVideoCall

Can be used to make a two party video call. It will return callID in the case of success otherwise failure error code will be return.

- **Format**

```
public void maxwellCreateTwoPartyVideoCall(
                        String calledURI,
                        int accountId,
                         RequestType request)
                        throws RemoteException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
calledURI	String	in	
accountId	int	in	
request	RequestType	in	

- **Return**
  None.

- **Exceptions**
  RemoteException.

4.2.20doDivertConfig

Can be used to change the call divert settings.

- **Format**

```
public void doDivertConfig(
                List<RedirectionData> list)
                throws RemoteException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
list	List<RedirectionData>	in	

- **Return**
  None.

- **Exceptions**
  RemoteException.

4.2.21addTwoPartyCallToConference

Can be used to add a active two party call into a active conference call. It will return callID in the case of success otherwise failure error code will be return.

- **Format**

```
public void addTwoPartyCallToConference(
                final int callIDTwoPartyCall,
                final int callIDconferenceCall,
                final RequestType request)
                throws RemoteException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
callIDTwoPartyCall	int	in	
callIDconferenceCall	int	in	
request	RequestType	in	

- **Return**
  None.

- **Exceptions**
  RemoteException.

4.2.22addNumberToConference

Can be used to add a number into a active conference call. It will return callID in the case of success otherwise failure error code will be return.

- **Format**

```
public void addNumberToConference(
                final String participantSipUri,
                final int callIDconferenceCall,
                RequestType request)
                throws RemoteException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
participantSipUri	String	in	
callIDconferenceCall	int	in	
request	RequestType	in	

- **Return**
  None.

- **Exceptions**
  RemoteException.

4.2.23holdConference

Can be used to hold a conference call. It will return success or failure code.

- **Format**

```
public void holdConference(
                int callIDconferenceCall,
                RequestType request)
                throws RemoteException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
callIDconferenceCall	int	in	
request	RequestType	in	

- **Return**

  None.

- **Exceptions**

  RemoteException.

4.2.24resumeConference

Can be used to resume a hold conference call. It will return success or failure code.

- **Format**

```
public void resumeConference(
                int callIDconferenceCall,
                RequestType request)
                throws RemoteException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
callIDconferenceCall	int	in	
request	RequestType	in	

- **Return**
  None.

- **Exceptions**
  RemoteException.

4.2.25swapConference

Can be used to swap between an active and hold conference. It will return success or failure code.

- **Format**

```
public void swapConference(
            final int activeCallId,
            final boolean isActiveCallConference,
            final int holdCallId,
            final boolean isHoldCallConference,
            final RequestType request)
            throws RemoteException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
activeCallId	int	in	
isActiveCall Conference	boolean		
holdCallId	int		
isHoldCallCo nference	boolean	in	
request	RequestTyp e		

- **Return**
  None.

- **Exceptions**
  RemoteException.

5 Audio APIs

5.1 Audio API List

Audio API list is described in Figure 5-1.

Figure 5-1 Audio API List

package: android.media

Class	Method	Contents	Comments
AudioManager	adjustStreamVolume	Adjusts the volume of an active stream by one step in a direction.	
	setStreamVolume	Sets the volume index for a particular stream.	
	getStreamMaxVolume	Returns the maximum volume index for a particular stream.	
	setStreamMute	Mute or unmute an audio stream.	
	setSpeakerphoneOn	Sets the speakerphone on or off	
	isSpeakerphoneOn	Checks whether the speakerphone is on or off.	
	setMicrophoneMute	Sets the microphone mute on or off.	
	isMicrophoneMute	Checks whether the microphone mute is on or off.	
	setMode	Sets the audio mode. Audio mode wil decide the routing of device	
	getMode()	Returns the current audio mode.	
	AudioTrack	Audio track is used to play the raw PCM data.	
	AudioRecord	Audio record is used to to record the raw PCM data	

5.2 AudioManager class

5.2.1 adjustStreamVolume

Adjusts the volume of an active stream by one step in a direction.

- **Format**

```
public void adjustStreamVolume(
          int streamType,
          int direction,
          int flags)
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
streamType	int	in	The stream type to adjust. One of STREAM_VOICE_CALL, STREAM_SYSTEM, STREAM_RING, STREAM_MUSIC or STREAM_ALARM
direction	int	in	The direction to adjust the volume. One of ADJUST_LOWER, ADJUST_RAISE, or ADJUST_SAME.
flags	int	in	One or more flags

- **Return**
  None.

- **Exceptions**
  None.

5.2.2 setStreamVolume

Sets the volume index for a particular stream.

- **Format**

```
public void setStreamVolume(
          int streamType,
          int index,
          int flags)
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
streamType	int	in	The stream whose volume index should be set.
index	int	in	The volume index to set. See getStreamMaxVolume(int) for the largest valid value.
flags	int	in	One or more flags.

- **Return**
  None.

- **Exceptions**
  None.

5.2.3 getStreamMaxVolume

Returns the maximum volume index for a particular stream.

- **Format**

```
public int getStreamMaxVolume(
          int streamType)
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
streamType	int	in	The stream type whose maximum volume index is returned.

- **Return**
  The maximum valid volume index for the stream.

- **Exceptions**
  None.

5.2.4 setStreamMute

Mute or unmute an audio stream.

- **Format**

```
public void setStreamMute(
          int streamType,
          boolean state)
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
streamType	int	in	The stream to be muted/unmuted.
state	boolean	in	The required mute state: true for mute ON, false for mute OFF

- **Return**
  None.

- **Exceptions**
  None.

5.2.5 setSpeakerphoneOn

Sets the speakerphone on or off.

- **Format**

```
public void setSpeakerphoneOn(
          boolean on)
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
on	boolean	in	set true to turn on speakerphone false to turn it off

- **Return**
  None.

- **Exceptions**
  None.

5.2.6 isSpeakerphoneOn

Checks whether the speakerphone is on or off.

- **Format**

```
public boolean isSpeakerphoneOn()
```

- **Parameters**
  None.

- **Return**
  True if speakerphone is on, false if it's off.

- **Exceptions**
  None.

5.2.7 setMicrophoneMute

Sets the microphone mute on or off.

- **Format**

```
public void setMicrophoneMute(
          boolean on)
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
on	boolean	in	Set true to mute the microphone and false to turn mute off

- **Return**
  None.

- **Exceptions**
  None.

5.2.8 isMicrophoneMute

Checks whether the microphone mute is on or off.

- **Format**

```
public boolean isMicrophoneMute()
```

- **Parameters**
  None.

- **Return**
  True if microphone is muted, false if it's not.

- **Exceptions**
  None.

5.2.9 setMode

Sets the audio mode. The audio mode encompasses audio routing and the behavior of the telephony layer. Informs the HAL about the current audio state so that it can route the audio appropriately.

- **Format**

```
public void setMode(
            int mode)
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
mode	int	in	the requested audio mode MODE_NORMAL, MODE_RINGTONE, MODE_IN_CALL, MODE_IN_COMMUNICATION

- **Return**
  None.

- **Exceptions**
  None.

5.2.10getMode

Returns the current audio mode.

- **Format**

```
public int getMode()
```

- **Parameters**
  None.

- **Return**
  Returns current audio mode which can be MODE_NORMAL, MODE_RINGTONE, MODE_IN_CALL, MODE_IN_COMMUNICATION from the HAL.

- **Exceptions**
  None.

5.2.11AudioTrack

Audio track is used to play the raw PCM data.

- **Format**

```
public AudioTrack(
          int streamType,
          int sampleRateInHz,
          int channelConfig,
          int audioFormat,
          int bufferSizeInBytes,
          int mode) throws
          IllegalArgumentException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
streamType	int	in	The type of the audio stream. See STREAM_VOICE_CALL, STREAM_SYSTEM, STREAM_RING, STREAM_MUSIC, STREAM_ALARM, STREAM_NOTIFICATION
sampleRateInHz	int	in	the sample rate expressed in Hertz
channelConfig	int	in	describes the configuration of the audio channels. See CHANNEL_OUT_MONO and CHANNEL_OUT_STEREO
audioFormat	int	in	the format in which the audio data is represented. See ENCODING_PCM_16BIT and ENCODING_PCM_8BIT

bufferSizeInBytes	int	in	the total size (in bytes) of the buffer where audio data is read from for playback. If using the AudioTrack in streaming mode, you can write data into this buffer in smaller chunks than this size. If using the AudioTrack in static mode, this is the maximum size of the sound that will be played for this instance. See getMinBufferSize(int, int, int) to determine the minimum required buffer size for the successful creation of an AudioTrack instance in streaming mode. Using values smaller than getMinBufferSize() will result in an initialization failure.
mode	int	in	streaming or static buffer. See MODE_STATIC and MODE_STREAM

- **Return**
  None

- **Exceptions**
  IllegalArgumentException.

5.2.12AudioRecord

Audio record is used to record the raw PCM data

- **Format**

```
public AudioRecord(
          int audioSource,
          int sampleRateInHz,
          int channelConfig,
          int audioFormat,
          int bufferSizeInBytes) throws
          IllegalArgumentException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
audioSource	int	in	The recording source. See MediaRecorder.AudioSource for recording source definitions.
sampleRateInHz	int	in	The sample rate expressed in Hertz. 44100Hz is currently the only rate that is guaranteed to work on all devices, but other rates such as 22050, 16000, and 11025 may work on some devices.
channelConfig	int	in	Describes the configuration of the audio channels. See CHANNEL_IN_MONO and CHANNEL_IN_STEREO. CHANNEL_IN_MONO is guaranteed to work on all devices.
audioFormat	int	in	The format in which the audio data is represented. See ENCODING_PCM_16BIT and ENCODING_PCM_8BIT

bufferSizeInBytes	int	in	The total size (in bytes) of the buffer where audio data is written to during the recording. New audio data can be read from this buffer in smaller chunks than this size. See getMinBufferSize(int, int, int) to determine the minimum required buffer size for the successful creation of an AudioRecord instance. Using values smaller than getMinBufferSize() will result in an initialization failure.

- **Return**
  None.

- **Exceptions**
  IllegalArgumentException.

6 Camera APIs

6.1 *Camera API List*

Camera API list is described in Figure 6-1.

Figure 6-1 Camera API List

package: android.hardware

Class	Method	Contents	Comments
Camera	getNumberOfCameras	Returns the number of physical cameras available on this device.	
	open	Creates a new Camera object to access a particular hardware camera	
	lock	Re-locks the camera to prevent other processes from accessing it. Camera objects are locked by default unless unlock() is called	
	unlock	Unlocks the camera to allow another process to access it. Normally, the camera is locked to the process with an active Camera object until release() is called.	
	startPreview	Starts capturing and drawing preview frames to the screen	
	stopPreview	Stops capturing and drawing preview frames to the surface, and resets the camera for a future call to startPreview().	

6.2 Camera class

6.2.1 getNumberOfCameras

Returns the number of physical cameras available on this device.

- **Format**

```
public static int getNumberOfCameras()
```

- **Parameters**
  None.

- **Return**
  Returns the number of physical cameras available on this device.

- **Exceptions**
  None.

6.2.2 open

Creates a new Camera object to access a particular hardware camera. If the same camera is opened by other applications, this will throw a RuntimeException.

- **Format**

```
static Camera open(
                int CameraID)
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
CameraID	int	in	The hardware camera to access, between 0 and getNumberOfCameras() -1.

- **Return**
  Returns opened camera handle.

- **Exceptions**
  RuntimeException.

6.2.3 lock

Re-locks the camera to prevent other processes from accessing it. Camera objects are locked by default unless unlock() is called.

- **Format**

```
public final void lock()
```

- **Parameters**
  None.

- **Return**
  None.

- **Exceptions**
  RuntimeException.

6.2.4 unlock

Unlocks the camera to allow another process to access it. Normally, the camera is locked to the process with an active Camera object until release() is called.

- **Format**

```
public final void unlock()
```

- **Parameters**
  None.

- **Return**
  None.

- **Exceptions**
  RuntimeException.

6.2.5 startPreview

Starts capturing and drawing preview frames to the screen. Preview will not actually start until a surface is supplied with setPreviewDisplay(SurfaceHolder) or setPreviewTexture(SurfaceTexture).

- **Format**

```
public final void startPreview()
```

- **Parameters**
  None.

- **Return**
  None.

- **Exceptions**
  None.

6.2.6 stopPreview

Stops capturing and drawing preview frames to the surface, and resets the camera for a future call to startPreview().

- **Format**

```
public final void stopPreview()
```

- **Parameters**
  None.

- **Return**
  None.

- **Exceptions**
  None.

7 Image Capture APIs

7.1 Image Capture API List

Image Capture API list is described in Figure 7-1.

Figure 7-1 Camera API List

package: android.hardware

Class	Method	Contents	Comments
Camera	takePicture	Triggers an asynchronous image capture. The camera service will initiate a series of callbacks to the application as the image capture progresses. The shutter callback occurs after the image is captured. This can be used to trigger a sound to let the user know that image has been captured. The raw callback occurs when the raw image data is available (NOTE: the data will be null if there is no raw image callback buffer available or the raw image callback buffer is not large enough to hold the raw image). The postview callback occurs when a scaled, fully processed postview image is available (NOTE: not all hardware supports this). The jpeg callback occurs when the compressed image is available. If the application does not need a particular callback, a null can be passed instead of a callback method.	

7.2 Camera class

7.2.1 takePicture

Triggers an asynchronous image capture. The camera service will initiate a series of callbacks to the application as the image capture progresses. The shutter callback occurs after the image is captured. This can be used to trigger a sound to let the user know that image has been captured. The raw callback occurs when the raw image data is available (NOTE: the data will be null if there is no raw image callback buffer available or the raw image callback buffer is not large enough to hold the raw image). The postview callback occurs when a scaled, fully processed postview image is available (NOTE: not all hardware supports this). The jpeg callback occurs when the compressed image is available. If the application does not need a particular callback, a null can be passed instead of a callback method.

- **Format**

```
public final void takePicture(
                ShutterCallback shutter,
                PictureCallback raw,
                PictureCallback postview,
                PictureCallback jpeg)
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
shutter	ShutterCallback	in	the callback for image capture moment, or null
raw	PictureCallback	in	the callback for raw (uncompressed) image data, or null
postview	PictureCallback	in	callback with postview image data, may be null
jpeg	PictureCallback	in	the callback for JPEG image data, or null

- **Return**

  None.

- **Exceptions**

  None.

8 Media Recorder APIs

8.1 Media Recorder API List

Media Recorder API list is described in Figure 8-1.

Figure 8-1 Media Recorder API List

package: android.media

Class	Method	Contents	Comments
MediaRecorder	setCamera	Sets a Camera to use for recording. Use this function to switch quickly between preview and capture mode without a teardown of the camera object.unlock() should be called before this.	
	setAudioSource	Sets the audio source to be used for recording. If this method is not called, the output file will not contain an audio track.	
	setVideoSource	Sets the video source to be used for recording. If this method is not called, the output file will not contain an video track.	
	setOutputFormat	Sets the format of the output file produced during recording. Call this after setAudioSource()/ setVideoSource() but before prepare().	
	setAudioEncoder	Sets the audio encoder to be used for recording. If this method is not called, the output file will not contain an audio track. Call this after setOutputFormat() but before prepare().	

	setVideoEncoder	Sets the video encoder to be used for recording. If this method is not called, the output file will not contain an video track. Call this after setOutputFormat() and before prepare().	
	setOutputFile	Sets the path of the output file to be produced. Call this after setOutputFormat() but before prepare().	
	prepare	Prepares the recorder to begin capturing and encoding data. This method must be called after setting up the desired audio and video sources, encoders, file format, etc., but before start().	
	start	Begins capturing and encoding data to the file specified with setOutputFile(). Call this after prepare().	
	stop	Stops recording. Call this after start(). Once recording is stopped, you will have to configure it again as if it has just been constructed.	

8.2 MediaRecorder class

8.2.1 setCamera

Sets a Camera to use for recording. Use this function to switch quickly between preview and capture mode without a teardown of the camera object. unlock() should be called before this. Must call before prepare().

- **Format**

```
public void setCamera(
          Camera c)
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
c	Camera	in	The Camera to use for recording

- **Return**
  None.

- **Exceptions**
  None.

8.2.2 setAudioSource

Sets the audio source to be used for recording. If this method is not called, the output file will not contain an audio track. The source needs to be specified before setting recording-parameters or encoders. Call this only before setOutputFormat().

- **Format**

```
public void setAudioSource(
          int audio_source)
          throws
          IllegalStateException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
audio_source	int	in	the audio source to use

- **Return**
  None.

- **Exceptions**
  IllegalStateException.

8.2.3 setVideoSource

Sets the video source to be used for recording. If this method is not called, the output file will not contain a video track. The source needs to be specified before setting recording-parameters or encoders. Call this only before setOutputFormat().

- **Format**

```
public void setVideoSource(
          int video_source)
          throws
          IllegalStateException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
video_source	int	in	the video source to use

- **Return**
  None.

- **Exceptions**
  IllegalStateException.

8.2.4 setOutputFormat

Sets the format of the output file produced during recording. Call this after setAudioSource()/setVideoSource() but before prepare().

- **Format**

```
public void setOutputFormat(
          int output_format)
          throws
          IllegalStateException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
output_format	int	in	the output format to use. The output format needs to be specified before setting recording-parameters or encoders.

- **Return**
  None.

- **Exceptions**
  IllegalStateException.

8.2.5 setAudioEncoder

Sets the audio encoder to be used for recording. If this method is not called, the output file will not contain an audio track. Call this after setOutputFormat() but before prepare().

- **Format**

```
public void setAudioEncoder(
          int audio_encoder)
          throws
          IllegalStateException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
audio_encoder	int	in	the audio encoder to use.

- **Return**
  None.

- **Exceptions**
  IllegalStateException.

8.2.6 setVideoEncoder

Sets the video encoder to be used for recording. If this method is not called, the output file will not contain an video track. Call this after setOutputFormat() and before prepare().

- **Format**

```
public void setVideoEncoder(
          int video_encoder)
          throws
          IllegalStateException
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
video_encoder	int	in	the video encoder to use.

- **Return**
  None.

- **Exceptions**
  IllegalStateException.

8.2.7 setOutputFile

Sets the path of the output file to be produced. Call this after setOutputFormat() but before prepare().

- **Format**

```
public void setOutputFile(
          String path)
```

- **Parameters**

Parameter	Type	IN/OUT	Comments
path	String	in	The pathname to use.

- **Return**
  None.

- **Exceptions**
  IllegalStateException.

8.2.8 prepare

Prepares the recorder to begin capturing and encoding data. This method must be called after setting up the desired audio and video sources, encoders, file format, etc., but before start().

- **Format**

```
public void prepare()
          throws
          IllegalStateException,
          IOException
```

- **Parameters**
  None.

- **Return**
  None.

- **Exceptions**
  IllegalStateException
  IOException

8.2.9 start

Begins capturing and encoding data to the file specified with setOutputFile(). Call this after prepare().

- **Format**

```
public void start()
            throws
            IllegalStateException
```

- **Parameters**
  None.

- **Return**
  None.

- **Exceptions**
  IllegalStateException.

8.2.10stop

Stops recording. Call this after start(). Once recording is stopped, you will have to configure it again as if it has just been constructed.

- **Format**

```
public void stop()
          throws
          IllegalStateException
```

- **Parameters**
  None.

- **Return**
  None.

- **Exceptions**
  IllegalStateException.