Gigaset Maxwell-10

3rd-Party SDK API Specification

June 2016, Ver. 2.1

Revision No.	Date	Author	Comments for revision
0.1	04/10/2013	R. Singhai	First release
0.2	08/10/2013	R. Singhai	Added table of contents with page numbers.
2.0	2015-12-23	J. Stahl	reworked for Android-5.1
2.1	2016-06-14	J. Stahl	minor modifications

Table of Contents

1. Introduction

1.1 Scope and Intention

This document lists the 3rd-party-SDK-API for the Gigaset Maxwell-10.

1.2 Acronyms and Abbreviations

Acronym / Abbreviation	Explanation
API	Application Programming Interface
SDK	Software Development Kit
AIDL	Android Interface Definition Language
DB	Data Base

2. Requirements and Constraints

References:

https://teamwork.gigaset.com/gigawiki/display/GPPPO/Maxwell+10

https://teamwork.gigaset.com/gigawiki/download/attachments/462193163/Gigaset%20Maxwell%2010%20APIs_for_3rd_parties%20v2.xls?version=2&modificationDate=1465906798245&api=v2

3. Interfaces

Interfaces are exposed from contacts (Maxwell-10 Directory application), call log (Maxwell Call list application) and Call Control (Maxwell-10 PhoneApp application) for 3rd party applications.

3.1 Contacts API's

Name of library avilable for 3rd party application development : DirectoryHelpder.jar
This library file provides APIs which can be used by any 3rd party applications to access contacts.

1 public static ContactsHelper getContactsHelperInstance(Context context)
Description:
Returns the Static instance of the Contacts Helper which can be used for accessing all other APIs

2 public List<ContactBean> getBasicContactsList(String limit) throws DirectoryHelperException,FileNotFoundException, IOException, Exception
Description:
Returns all available contacts in the DB

3 public ContactBean getContact(int contactId) throws DirectoryHelperException
Description:
Returns a single contact using the contact ID

4 public ContactBean getContact(ContactBean rowContact) throws DirectoryHelperException
Description:
Returns a single contact using the contact bean object

5 public BasicContact getBasicContactDetails(String phoneNumber) throws DirectoryHelperException
Description:
Returns a single contact using the phone number of the contact

6 public void getBasicContactDetails(String phoneNumber, ContactListener contactListener) throws DirectoryHelperException
Description:
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.

7 public String getcompName(int contactId)
Description:
Returns the Company Name of a contact using the contactId

8 public boolean insertContact(ContactBean contactBean) throws DirectoryHelperException
Description:
Inserts a new Contacts into the Contacst DB

9 public boolean updateContact(ContactBean contactBean) throws DirectoryHelperException
Description:
Updates a contacts into the ContactsDB

10 public boolean deleteAContact(int rawId) throws DirectoryHelperException
Description:
deletes a contacts from the DB

11 public Uri getPictureUri(String phoneNumber)
Description:
Returns the Picture URI of the contacts if present

12 public AsyncRequestID connectAndSearch(String text, SearchResultListener context, String type) throws LDAPException, Exception
Description:
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)

13 public void searchContact(String searchText, String directoryType)
Description:
Searches a contact using the string in the XML net directory, results are returned using the call back public void onContactLoaded(final ContactBean contact)

3.2 Call log APIs – only local

Name of library avilable for 3rd party application development : DirectoryHelpder.jar
This library file provides APIs which can be used by any 3rd party applications to access call lists/logs

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

2 public void deleteSingleCallLog(CallInfo callinfo);
Description:
Deletes a single call log from DB, status is returned using the call back method public void onDeleteComplete(CallLogList result);

3 public void deleteMultipleCallLogs(List<String> callIds);
Description:
Deletes a list of call logs from the DB, status is returned using the call back method public void onDeleteComplete(CallLogList result);

4 public Uri insertCallLogDetails(BasicCallInfo callLog) throws DirectoryHelperException;
Description:
Insert a call log into DB

3.3 Call control APIs (Maxwell Phone app)

Allows a user/application to do VoIP call specific functions. There are two ways to use the services exposed by phone app
1. Intent - Standard usage of Intent to make the call.
2. SipService - Exposed different interfaces via AIDL. MaxwellPhone app has implemented an SipService.java, other application can bind with this service and call different APIs. This should be exclusive to Maxwell application only.

Intent:

1 Intent "android.intent.action.CALL"

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

i.e.

```
Intent callIntent = new Intent(Intent.ACTION_CALL);

callIntent.setData(Uri.parse(""tel:"" + phoneNum));

startActivity(callIntent);
```

API's:

Important note: MaxwellPhone app has implemented an SipService.java, other application can bind with this service and call different APIs. This should be exclusive to Maxwell application only.

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

Description:
Register a ResultReceiver for getting the callback.

2 public void unregisterReceiver(String className) throws RemoteException

Description:
Unregister a ResultReceiver.

3 public void initSipStack(final boolean isCheckSyncEnable, RequestType request)
throws RemoteException
Description:
Used to initialize the doubango SIP stack. Registering audio and video producer consumer.

4 public void setConfig(final RequestType request)
throws RemoteException
Description:
Used to set the configuration in doubango stack. It will retrun success or failure code.

5 public void maxwellCreateTwoPartyCall(String calledURI, int accountId, final String requestType,RequestType request)
throws RemoteException
Description:
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.

6 public void hangUp(int callId, RequestType request)
throws RemoteException
Description:
Can be used to hangUp a call. It will retrun success or failure code.

7 public void holdCall(int callId, RequestType request)
throws RemoteException
Description:
Can be used to hold a call. It will return success or failure code.

8 public void resumeCall(int callId, RequestType request)
throws RemoteException
Description:
Can be used to resume a hold call. It will return success or failure code.

9 public void acceptCall(int callId, RequestType request)
throws RemoteException
Description:
Can be used to accept a incoming call. It will return success or failure code.

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

Description:

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

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

Description:

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

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

Description:

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

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

Description:

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

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

Description:

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

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

Description:

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

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

Description:

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

17 public void createConfernceFromUri(final int activeCallId, final String number, final RequestType request)
throws RemoteException
Description:
Can be used to create a conference call from a active call and Uri. It will return success or failure code.

18 public void hangUpConference(int confId, RequestType request)
throws RemoteException
Description:
Can be used to hangup a conference. It will return success or failure code.

19 public void maxwellCreateTwoPartyVideoCall(String calledURI, int accountId, final String requestType , RequestType request)
throws RemoteException
Description:
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.

20 public void doDivertConfig(List<RedirectionData> list)
throws RemoteException
Description:
Can be used to change the call divert settings.

21 public void addTwoPartyCallToConference(final int callIDTwoPartyCall, final int callIDconferenceCall, final RequestType request)
throws RemoteException
Description:
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.

22 public void addNumberToConference(final String participantSipUri, final int callIDconferenceCall, RequestType request)
throws RemoteException
Description:
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 returned.

23 public void holdConference(int callIDconferenceCall, RequestType request)
throws RemoteException
Description:
Can be used to hold a conference call. It will return success or failure code.

24 public void resumeConference(int callIDconferenceCall, RequestType request)
throws RemoteException
Description:
Can be used to resume a hold conference call. It will return success or failure code.

25 public void swapConference(final int activeCallId, final boolean isActiveCallConference, final int holdCallId, final boolean isHoldCallConference, final RequestType request)
throws RemoteException
Description:
Can be used to swap between an active and hold conference. It will return success or failure code.

26 void swapCall(int activeCallId, int holdCallId, in RequestType request)
Description:
To swap an active call with hold call.
param activeCallId: callId of active call to be swap
param holdCallId: callId of hole call to be swap
param requestType: type of request

27 void stopSipgateCallRecord(int callId, in RequestType request)
Description:
To stop “sipgate” call record

28 void startSipgateCallRecord(int callId, in RequestType request)
Description:
To start “sipgate” call record

29 void dectEventActionWithCodec(int handsetCallId, int callId, int actionTodo, boolean isWideBand)
Description:
To do dect event action
param handsetCallId: call id on dect handset
param callId: on which call dect event to perform
param actionTodo: which action to perform
param isWideBand: if to enable wide band codec

30 void dectEventAction(int handsetCallId, int callId, int actionTodo)

Description:
To do dect event action
param handsetCallId: call id on dect handset
param callId: on which call dect event to perform
param actionTodo: which action to perform

31 void transferIncomingCall(int incomingCallId, in String tranferToNumber,in RequestType request);
throws RemoteException

Description:
To transfer directly incoming call

32 void recordOnServer(int callId, boolean isRecord)
throws RemoteException

Description:
To start stop record call on server
param callId : call id
param isRecord: true to start record and false to stop record