android-rcs-ims-stack

RCS / IMS stack for Android platform

RCS API Specification

Edition 1.0

Author:
Jean-Marc AUFFRET
jeanmarc.auffret@orange-ftgroup.com

13 pages

CONTENTS

FIGURES

1. Introduction

1.1. Purpose of the document

This document is a detailed specification of the RCS 2.0 API.

The goal of the RCS API is to offer a high level Java API to implement RCS applications (e.g. rich address book, rich call telephony application, chat view, RCS widget, etc).

The RCS API offers the following API:

- Presence API (presence sharing, presence subscription & publishing, anonymous fetch).
- Rich call API (capabilities, image sharing, video sharing).
- Rich messaging API (file transfer, chat, .etc).

1.2. Reference documents

N°	Title	Release
1	RCS specification release documents: http://www.gsmworld.com/our-work/mobile_lifestyle/rcs/rcs_specification_documents.htm	06/2009
2	GSMA IR.74 - Video Share Interoperability Specification	21/03/2007
3	GSMA IR.79 - Image Sharing Interoperability Specification	28/01/2008

1.3. Abbreviations

Abbreviation	Name
IMS	IP Multimedia Subsystem
CSh	Content sharing
EAB	Enhanced Address Book
SIP	Session Initiation Protocol
RTP	Real Time Protocol
MSRP	Media Session Relay Protocol
AIDL	Android Inter-process communication protocol

1.4. Terminology

Term	Description
Activity	Android UI application
Service	Android background process
Intent	Android description of an operation to be performed
Intent filter	Structured description of an intent to be matched
Broadcast receiver	An application class that listens for Intents that are broadcasted

1.5. Targeted OS release

The targeted release is Android 2.x.

2. RCS API definition

2.1. Common API definition

The RCS API uses the following Android concepts:

- Intents mechanism to broadcast incoming events (e.g. notification) and incoming invitations to any Android activity or broadcast receiver declared in the device.
- AIDL interfaces to initiate and to manage session in real time (start, session monitoring, stop). Session events are managed thanks to callback mechanism.

The RCS API throws an exception if the IMS core layer is not initialized or not registered to the IMS platform.

Note: Remote application exceptions are not yet supported by the AIDL SDK, a generic AIDL exception is thrown instead.

How to use Intents?

By dynamically registering an instance of a class with `Context.registerReceiver()` or by using the `<receiver>` tag in your `AndroidManifest.xml`. Then the type of requested event is fixed in the Intent filter associated to the receiver, each RCS API has its own list of available intents.

How to use the AIDL interface?

The AIDL interface is hidden by a static interface which main goal is to connect to the AIDL server side and to monitor the connection with it.

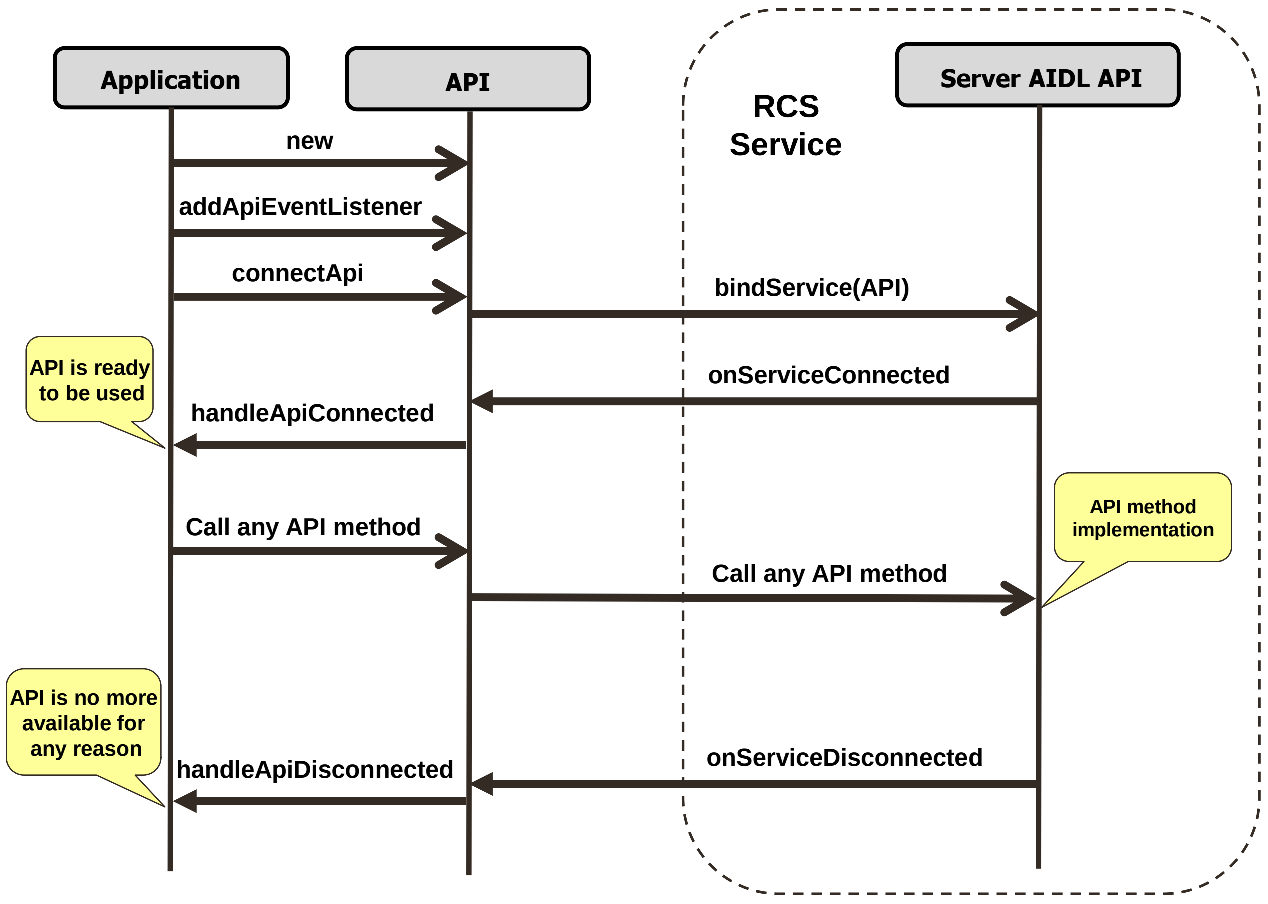

Figure 1: Common API architecture

Note: an exception is thrown if the API is not yet initialized when calling a method.

2.2. Presence API

2.2.1. Description

The presence API contains:

- An AIDL interface used to publish presence info and to manage presence sharing with other contacts.
- Intents which permit to broadcast new presence info and new status notifications.

2.2.1. Presence info document

The presence info document contains the following attributes:

- Timestamp.
- Presence status: Offline | Online.
- Free text.
- Favorite link: link, name.
- Photo-icon: mime-type, width, height, etag.
- Hyperavailability status: Boolean.
- Supported capabilities: video sharing, image sharing, file transfer, IM session, CS video.
- Geoloc info: latitude, longitude, altitude.

2.2.2. AIDL interface

```
// Set my presence info:
// - Returns a Boolean result
// - Throws an exception if API not initialized or IMS core not ready
boolean setMyPresenceInfo(in PresenceInfo info)

// Set my hyper-availability status:
// - Returns a Boolean result
// - Throws an exception if API not initialized or IMS core not registered
boolean setMyHyperAvailabilityStatus(in boolean status)

// Get the hyper-availability expiration time:
// - Returns an expiration time in milliseconds
// - Throws an exception if API not initialized or IMS core not ready
long getHyperAvailabilityExpiration()

// Invite a contact to share its presence:
// - Returns a Boolean result
// - Throws an exception if API not initialized or IMS core not ready
boolean inviteContact(in String contact)

// Accept the sharing invitation from a contact:
// - Returns a Boolean result
// - Throws an exception if API not initialized or IMS core not ready
boolean acceptSharingInvitation(in String contact)

// Reject the sharing invitation from a contact:
// - Returns a Boolean result
// - Throws an exception if API not initialized or IMS core not ready
boolean rejectSharingInvitation(in String contact)

// Ignore the sharing invitation from a contact:
// - Returns a Boolean result
// - Throws an exception if API not initialized or IMS core not ready
void ignoreSharingInvitation(in String contact)

// Revoke a shared contact:
// - Returns a Boolean result
// - Throws an exception if API not initialized or IMS core not ready
boolean revokeContact(in String contact)

// Unrevoke a revoked contact:
// - Returns a Boolean result
// - Throws an exception if API not initialized or IMS core not ready
boolean unrevokeContact(in String contact)

// Unblock a blocked contact:
// - Returns a Boolean result
// - Throws an exception if API not initialized or IMS core not ready
boolean unblockContact(in String contact)

// Get the list of granted contacts:
// - Returns a list of Tel-URI as String
// - Throws an exception if API not initialized or IMS core not ready
List<String> getGrantedContacts()

// Get the list of revoked contacts:
// - Returns a list of Tel-URI as String
// - Throws an exception if API not initialized or IMS core not ready
List<String> getRevokedContacts()

// Get the list of blocked contacts:
// - Returns a list of Tel-URI as String
// - Throws an exception if API not initialized or IMS core not ready
List<String> getBlockedContacts()

// Request capabilities for a contact (i.e anonymous fetch) in background:
// - The result is returned via an Intent
// - Throws an exception if API not initialized or IMS core not ready
void requestCapabilities(in String contact)
```

2.2.3. Intents declaration

```
// Intent broadcasted when a presence sharing invitation has been received,
// with parameters:
// - "contact": remote contact.
com.orangelabs.rcs.presence.PRESENCE_SHARING_INVITATION

// Intent broadcasted when user presence info has changed
com.orangelabs.rcs.presence.MY_PRESENCE_INFO_CHANGED

// Intent broadcasted when user presence status has changed,
// with parameters:
// - "status": hyper-availability status.
// - "expireAt": hyper-availability expiration time in milliseconds.
com.orangelabs.rcs.presence.MY_PRESENCE_STATUS_CHANGED

// Intent broadcasted when a contact info has changed,
// with parameters:
// - "contact": remote contact.
com.orangelabs.rcs.presence.CONTACT_INFO_CHANGED

// Intent broadcasted when a contact photo-icon has changed,
// with parameters:
// - "contact": remote contact.
com.orangelabs.rcs.presence.CONTACT_PHOTO_CHANGED

// Intent broadcasted when a presence sharing info has changed,
// with parameters:
// - "contact": remote contact.
// - "status": sharing status.
// - "reason": reason associated to the status.
com.orangelabs.rcs.presence.PRESENCE_SHARING_CHANGED

// Intent broadcasted when contact capabilities have been received (i.e.
// anonymous fetch),
// with parameters:
// - "contact": remote contact.
// - "org.gsma.videoshare": video sharing capability.
// - "org.gsma.imageshare": image sharing capability.
// - "org.openmobilealliance:File-Transfer": file transfer capability.
// - "org.openmobilealliance:IM-Session": IM session capability.
// - "org.3gpp.cs-videotelephony": CS video capability.
com.orangelabs.rcs.presence. CONTACT_CAPABILITIES
```

2.3. Rich call API

2.3.1. Description

The rich call API contains:

- An AIDL interface used to initiate content sharing session and to monitor a session (accept, reject, cancel, abort).
- Intents which permit to broadcast new session invitations and capabilities events.

Supported video encodings are H.263+ and H.264 in low quality (QCIF) and high quality (QVGA).

2.3.2. AIDL interface

```
// Request content sharing capabilities from a contact:
// - The result is returned via an Intent
// - Throws an exception if API not initialized or IMS core not registered
void requestContentSharingCapabilities(in String contact)

// Initiate a live video sharing session with a contact:
// - Returns a video sharing session
// - Throws an exception if API not initialized or IMS core not registered
IVideoSharingSession initiateLiveVideoSharing(in String contact, in IMediaPlayer
player)

// Initiate a pre-recorded video sharing session with a contact:
// - Returns a video sharing session
// - Throws an exception if API not initialized or IMS core not registered
IVideoSharingSession initiateVideoSharing(in String contact, in String file, in
IMediaPlayer player)

// Get a video sharing session from its session ID:
// - Returns a video sharing session or null if not found
IVideoSharingSession getVideoSharingSession(in String id)

// Initiate an image sharing session with a contact:
// - Returns an image sharing session
// - Throws an exception if API not initialized or IMS core not registered
IImageSharingSession initiateImageSharing(in String contact, in String file)

// Get an image sharing session from its session ID:
// - Returns an image sharing session or null if not found
IImageSharingSession getImageSharingSession(in String id)

// Image sharing session
interface IImageSharingSession {
    // Get session ID
    String getSessionID();

    // Get remote contact
    String getRemoteContact();

    // Get filename
    String getFilename();

    // Get filesize
    long getFilesize();

    // Accept the session invitation
    void acceptSession();

    // Reject the session invitation
    void rejectSession();

    // Cancel the session
    void cancelSession();

    // Add session listener
    void addSessionListener(in IImageSharingEventListener listener);

    // Remove session listener
    void removeSessionListener(in IImageSharingEventListener listener);
}

// Image sharing event listener
interface IImageSharingEventListener {
    // Session is started
    void handleSessionStarted();

    // Session has been aborted
    void handleSessionAborted();

    // Session has been terminated
    void handleSessionTerminated();

    // Session has been terminated by remote
    void handleSessionTerminatedByRemote();

    // Content sharing progress
    void handleSharingProgress(in long currentSize, in long totalSize);

    // Content sharing error
    void handleSharingError(in int error);

    // Image has been transferred
    void handleImageTransfered(in String filename);
}

// Video sharing session
interface IVideoSharingSession {
    // Get session ID
    String getSessionID();

    // Get remote contact
    String getRemoteContact();

    // Accept the session invitation
    void acceptSession();

    // Reject the session invitation
    void rejectSession();

    // Cancel the session
    void cancelSession();

    // Set the media renderer (only used for incoming session)
    void setMediaRenderer(in IMediaRenderer renderer);

    // Add session listener
    void addSessionListener(in IVideoSharingEventListener listener);

    // Remove session listener
    void removeSessionListener(in IVideoSharingEventListener listener);
}

// Video sharing event listener
interface IVideoSharingEventListener {

    // Session is started
    void handleSessionStarted();

    // Session has been aborted
    void handleSessionAborted();

    // Session has been terminated
    void handleSessionTerminated();

    // Session has been terminated by remote
    void handleSessionTerminatedByRemote();

    // Content sharing error
    void handleSharingError(in int error);
}
```

2.3.3. Intents declaration

```
// Intent broadcasted when content sharing capabilities have been exchanged,
// with parameters:
// - "contact": remote contact.
// - "image": image sharing capability.
// - "video": video sharing capability.
com.orangelabs.rcs.richcall.CONTENT_SHARING_CAPABILITIES

// Intent broadcasted when a new image sharing invitation has been received,
// with parameters:
// - "contact": remote contact.
// - "sessionId": session ID of the incoming session.
com.orangelabs.rcs.richcall.IMAGE_SHARING_INVITATION

// Intent broadcasted when a new video sharing invitation has been received,
// with parameters:
// - "contact": remote contact.
// - "sessionId": session ID of the incoming session.
com.orangelabs.rcs.richcall.VIDEO_SHARING_INVITATION
```

2.4. Rich messaging API

2.4.1. Description

The rich messaging API contains:

- An AIDL interface used to initiate sessions (IM, chat and file transfer).
- Intents used to receive incoming session invitation (IM, chat and file transfer).

2.4.2. AIDL interface

```
// Send an IM in short mode to a contact:
void sendShortIM(in String contact, in String txt)

// Send an IM in large mode to a contact:
// - Returns an IM session
IInstantMessageSession sendLargeIM(in String contact, in String txt)

// Transfer a file to a contact:
// - Returns a file transfer session
IFileTransferSession transferFile(in String contact, in String file)

// Get the file transfer session from its session ID
// - Returns a file transfer session or null if not found
IFileTransferSession getFileTransferSession(in String id)

// Initiate a one-to-one chat session with a contact:
// - Returns a chat session
IChatSession initiateOne2OneChatSession(in String contact, in String subject)

// Initiate an ad-hoc group chat session with a list of participants:
// - Returns a chat session
IChatSession initiateAdhocGroupChatSession(in String subject, in List<String>
participants)

// Initiate a conference group chat session:
// - Returns a chat session
IChatSession initiateAdhocGroupChatSession(in String subject, in String
conferenceID)

// Get a chat session from its session ID
// - Returns a chat session or null if not found
IChatSession getChatSession(in String id)

// File transfer session
interface IFileTransferSession {
   // Get session ID
   String getSessionID();

   // Accept the session invitation
   void acceptSession();

   // Reject the session invitation
   void rejectSession();

   // Cancel the session
   void cancelSession();

   // Add session listener
   void addSessionListener(in IFileTransferEventListener listener);

   // Remove session listener
   void removeSessionListener(in IFileTransferEventListener listener);
}

// File transfer event listener
interface IFileTransferEventListener {
   // Session is started
```

```
    void handleSessionStarted();

    // Session has been aborted
    void handleSessionAborted();

    // Session has been terminated
    void handleSessionTerminated();

    // Session has been terminated by remote
    void handleSessionTerminatedByRemote();

    // Data transfer progress
    void handleTransferProgress(in long currentSize, in long totalSize);

    // Transfer error
    void handleTransferError(in int error);

    // File has been transfered
    void handleFileTransfered(in String filename);
}

// Chat session interface
interface IChatSession {
    // Get session ID
    String getSessionID();

    // Get remote contact
    String getRemoteContact();

    // Get subject
    String getSubject();

    // Accept the session invitation
    void acceptSession();

    // Reject the session invitation
    void rejectSession();

    // Cancel the session
    void cancelSession();

    // Send a text message
    void sendMessage(in String text);

    // Set the is composing status
    void setIsComposingStatus(in boolean status);

    // Add session listener
    void addSessionListener(in IChatEventListener listener);

    // Remove session listener
    void removeSessionListener(in IChatEventListener listener);
}

// Chat event listener
interface IChatEventListener {
    // Session is started
    void handleSessionStarted();

    // Session has been aborted
    void handleSessionAborted();

    // Session has been terminated
    void handleSessionTerminated();

    // Session has been terminated by remote
    void handleSessionTerminatedByRemote();

    // New text message received
    void handleReceiveMessage(in String contact, in String text);

    // Is composing event
    void handleIsComposingEvent(in String contact, in boolean status);

    // Chat error
    void handleImError(in int error);
}
```

2.4.3. Intents declaration

```
// Intent broadcasted when a new IM has been received,
// with parameters:
// - "contact": remote contact.
// - "message": text plain message.
// - "receiveAt": date of reception.
com.orangelabs.rcs.messaging.IM

// Intent broadcasted when a new file transfer invitation has been received,
// with parameters:
// - "contact": remote contact.
// - "sessionId": session ID of the incoming session.
com.orangelabs.rcs.messaging.FILE_TRANSFER_INVITATION

// Intent broadcasted when a new chat invitation has been received,
// with parameters:
// - "contact": remote contact.
// - "subject": subject of the conference.
// - "sessionId": session ID of the incoming session.
com.orangelabs.rcs.messaging.CHAT_INVITATION
```