# Solaiemes RCS Solution Gateway

# OMA API Description

| | |
|---|---|
| **DocVersion:** | 1.1 |
| **Type:** | Tech Description |
| **Date:** | 05/10/2012 |

## REVISION

| Version | Date | Author | Comments |
|---|---|---|---|
| 1 | 13/09/2012 | David Culebras | Version 0.1 DRAFT<br>OMA API document |
| 2 | 21/09/2012 | David Culebras<br>Hugo Galán | Version 1.0<br>OMA API document updates |
| 3 | 05/10/2012 | Sergio López | Version 1.1<br>Added notifications |

# INDEX

## 1. Abstract

This document describes the OMA API interfaces provided by the Solaiemes RCS Solution Gateway. The API instantiates and controls a virtual RCS client impersonating a service on top of a RCS-enabled core.

Basic working knowledge of REST APIs is assumed, some very basic examples are provided in the last sections to help understanding API usage.

The APIs described in this document correspond to the Solaiemes RCS Solution Gateway. Security mechanisms, authorization, end-user confidentiality and other services intended to be customized for specific deployments are left out of the scope.

## 2. What is RCS-e / joyn

RCS-e is the standard of Rich Communication Suite (RCS) which will enable mobile phone end users to use instant messaging (IM), live video sharing and file share across any mobile phone on any network operator.

GSMA offers detailed information about this initiative, and it can be found:

http://www.gsma.com/rcs/

The RCS-e specs can be found on:

http://www.gsma.com/rcs/specifications/rcs-e-specifications/

joyn™ is the customer-facing brand to identify and promote the RCS services. joyn makes everyday, mobile to mobile communications more engaging. joyn brings you closer to the people in your mobile address book by combining all the ways you want to be in touch - Contacts, Chat, File share and Video share. Turn an everyday exchange into something much richer as you:

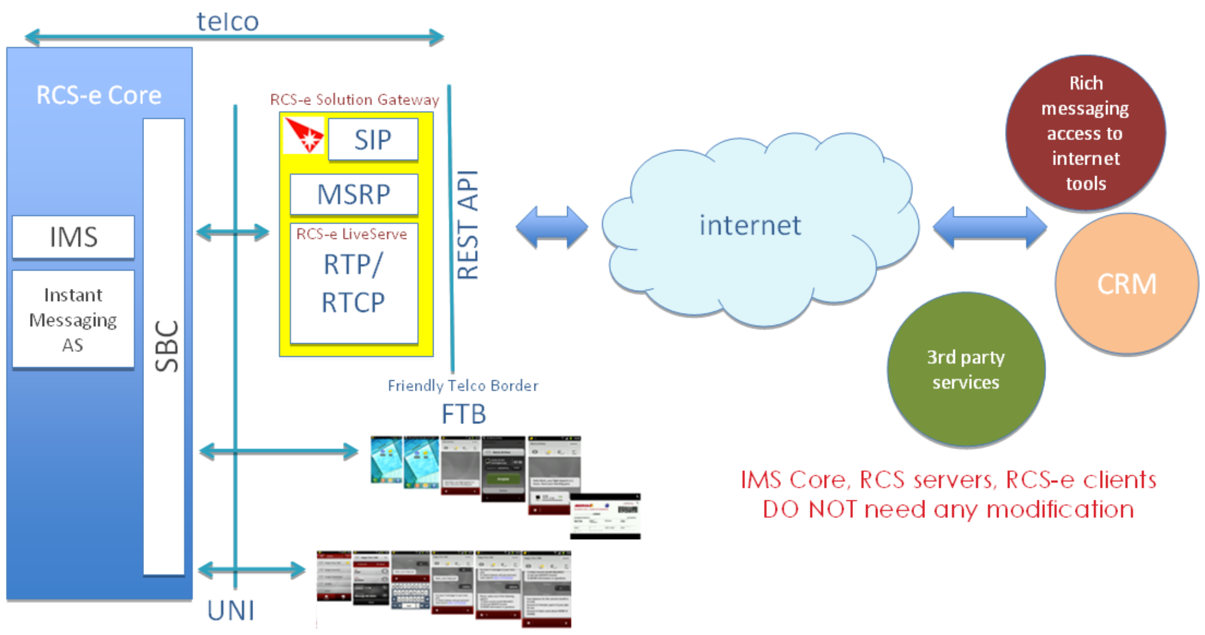
- Share the chat - with one-to-one and group instant messaging.
- Share the moment - enhance calls and chat by adding in videos, pictures, music and files.

## 3. RCS Solution Gateway description and features

The RCS Solution Gateway allows deploying services on top of RCS by creating virtual clients following the UNI approach (User to Network Interface), i.e., coming through the SBC to the network core in the same way that a mobile or desktop client does. Those virtual clients may be used as services for different purposes like virtual assistant, an alternative access to Internet services (mashing up their APIs), customer care/CRM, social media connector, etc.

Developers use the API provided by the RCS Solution Gateway, without needing to know about RCS technical details or telco protocols (SIP, XDM, MSRP, etc), taking advantage of RCS characteristics in their applications and solutions for enterprise, customer care, mass market apps, social, gaming, etc.

## 3.1. RCS Solution Gateway Architecture



Solaiemes RCS Solution Gateway is the fastest go-to-market solution for enhanced person-to-service use cases, empowering RCS offer and easing mass adoption.

Solaiemes RCS Solution Gateway instantiates virtual RCS service-clients supporting RCS-e features, and they can be used using the different API provided on next chapter.

# 4. OMA (Open Mobile Alliance) API

The OMA API program provides standardized interfaces to the service infrastructure residing within communication networks and on devices. Focused primarily between the service access layer and generic network capabilities, OMA API specifications allow operators and other service providers to expose device capabilities and network resources in an open and programmable way—to any developer community independent of the development platform.

Further information is available on OMA website, http://www.openmobilealliance.org

OMA API specs can be found here: http://www.openmobilealliance.org/api/APIDetails.aspx?ID=124

On this chapter we provide the resource URL to allow managing the different RCS operations.

## 4.1. Login / Logout

### 4.1.1.Create a new subscription to session notifications

This operation is used to create a new subscription to receive session notifications.

HTTP POST http://{serverRoot}/register/{apiVersion}/{userId}/subscriptions

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user who wants to subscribe to session notifications

**Body:**

SessionSubscription data structure

**Response:**

200 with a modified SessionSubscription data structure in the body or a 4xx code if an error happened.

### 4.1.2. Delete a subscription to session notifications

This operation is used to cancel the subscription to this notifications.

HTTP DELETE http://{serverRoot}/register/{apiVersion}/{userId}/subscriptions/{subscriptionId}

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user

**subscriptionId:** Identifier of the subscription

**Response:**

204 code to successful registers or a 4xx code to unsuccessful registers.

### 4.1.3. Read an individual chat notification subscription

This operation is used to retrieve information about a chat subscription.

HTTP GET http://{serverRoot}/register/{apiVersion}/{userId}/subscriptions/{subscriptionId}

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user

**subscriptionId:** Identifier of the subscription

**Response:**

200 with a SessionSubscription data structure in the body or a 4xx code if an error happened.

### 4.1.4. Read the list of active chat notification subscriptions

This operation is used to retrieve information about all active chat subscriptions (only one subscription for this resource can be created in this API version).

HTTP GET http://{serverRoot}/register/{apiVersion}/{userId}/subscriptions

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user

**subscriptionId:** Identifier of the subscription

**Response:**

200 with a SessionSubscriptionList data structure in the body or a 4xx code if an error happened.

### 4.1.5.Register

This operation logs the virtual RCS user into RCS Core; it can be understood as the operation to deploy the application. It should be invoked only once in an application lifetime.

A SIP REGISTER request is sent to the RCS Core, between other actions. A result code equals to 0 means that the REGISTER was successfully sent. Any other result code means that there was an error sending the REGISTER.

In any case, this result code only refers to the REGISTER request, not the IMS core response.

If application wants to know if registration with the service provider core was successful, it should wait for the registerSucces or registerError notification.

If the user is RCS 2.0, all the subscriptions and publications will be started after the registerSuccess event. Otherwise (the user is RCS-e), the OPTIONS polling will be started.

HTTP POST http://{serverRoot}/register/{apiVersion}/{userId}/sessions

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user to be registered

**Response:**

204 code to successful registers or a 4xx code to unsuccessful registers.

### 4.1.6.Unregister

This operation logs the virtual RCS user out from the RCS Core, it can be understood as the operation to un-deploy the application. Once is completed, the application is no longer connected to the RCS core. It should be invoked only once in an application lifetime.

As with the register operation, a REGISTER request is sent to the RCS Core. A result code equals to 0 means that it was successfully sent and any other result code means that there was an error sending it.

To know the complete result of the unregister process we will have to wait for the unregisterSuccess or unregisterError notification (see Error: Reference source not found for further detail).

If the user is RCS 2.0 all active subscriptions and publications will be terminated and RCS 2.0 clients watching the virtual user presence would be notified as per RCS core configuration and policies. If the user is RCS-e the OPTIONS polling will be terminated and RCS-e clients will notice when they try to send a new OPTIONS to the unregistered user.

HTTP DELETE http://{serverRoot}/register/{apiVersion}/{userId}/sessions

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user to be unregistered

**Response:**

204 code to successful registers or a 4xx code to unsuccessful registers.

## 4.2. Address book

### 4.2.1. Create a new subscription for address book changes

This operation is used to create a subscription in order to receive address book notifications.

HTTP POST http://{serverRoot}/addressbook/{apiVersion}/{userId}/subscriptions/abChanges

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user

**Body:**

AbChangesSubscription data structure

**Response:**

200 with a modified AbChangesSubscription data structure in the body or a 4xx code if an error happened.

### 4.2.2. Update/Extend the duration of the subscription

This operation is used to update or extend the duration of a previous subscription.

HTTP PUT http://{serverRoot}/addressbook/{apiVersion}/{userId}/subscriptions/abChanges/{subscriptionId}

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user

**subscriptionId:** Identifier of the subscription that will be update

**Body:**

AbChangesSubscription data structure

**Response:**

200 with a modified AbChangesSubscription data structure in the body or a 4xx code if an error happened.

### 4.2.3. Delete a subscription

This operation is used to delete a subscription.

HTTP DELETE http://{serverRoot}/addressbook/{apiVersion}/{userId}/subscriptions/abChanges/{subscriptionId}

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user

**subscriptionId:** Identifier of the subscription that will be delete

**Response:**

204 code to successful calls or a 4xx code to unsuccessful calls.

### 4.2.4. Retrieve information about individual subscription

This operation is used to retrieve information about a subscription for address book changes.

HTTP GET http://{serverRoot}/addressbook/{apiVersion}/{userId}/subscriptions/abChanges/{subscriptionId}

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user

**subscriptionId:** Identifier of the subscription

**Response:**

200 with an AbChangesSubscription data structure in the body or a 4xx code if an error happened.

### 4.2.5. Retrieve information about all active subscriptions

This operation is used to retrieve information about all subscriptions for this resource (only one subscription for this resource can be created in this API version).

HTTP GET http://{serverRoot}/addressbook/{apiVersion}/{userId}/subscriptions/abChanges

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user

**Response:**

200 with an AbChangesSubscriptionCollection data structure in the body or a 4xx code if an error happened.

### 4.2.6. Get contact list

Get the up-to-date list of contacts of a user. Every contact has information like the uri (sip or tel), the displayname or the capabilities it has.

To force a contact update we will use the getContact method in this service.

HTTP GET http://{serverRoot}/addressbook/{apiVersion}/{userId}/contacts

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user whose contacts are to be retrieved

**Response:**

200 with a ContactCollection data structure in the body or a 4xx code if an error happened.

### 4.2.7. Get contact

Get the detailed information of a contact. This detailed information is the contact's uri, displayname and last capabilities received. If the contact's capabilities are not updated when the getContact action is received, then the result will contain a set of capabilities that will be out of date. However an OPTIONS request to the contact will be sent and the updated capabilities will be available to check with a new getContact and a contactChanged notification will be sent to us (see Error: Reference source not found for further details).

HTTP GET http://{serverRoot}/addressbook/{apiVersion}/{userId}/contacts/{contactId}

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user whose contacts are to be retrieved

**contactId:** Identifier (tel or sip) of the contact we want to retrieve.

**Response:**

200 with a Contact data structure in the body or a 4xx code if an error happened.

### 4.2.8. Add / Edit contact

Add/Edit a contact to the contact list of the selected user. Once the invocation is finished we can check the result to determinate if the contact was successfully added / edited and we can wait for a contactAdded notification (see Error: Reference source not found) or we can check the result code to determinate if the contact was successfully edited.

HTTP PUT http://{serverRoot}/addressbook/{apiVersion}/{userId}/contacts/{contactId}

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user who wants to add a contact

**contactId:** Identifier (sip or tel) identifying the contact to be added

**Body:**

Contact data structure

**Response:**

200 with a modified Contact data structure in the body or a 4xx code if an error happened.

### 4.2.9. Remove contact

Remove a contact from the RCS contact list of the selected user. Once the invocation is finished we can check the result code to determinate if the contact was successfully removed or we can wait for a contactRemoved notification (see Error: Reference source not found).

HTTP DELETE http://{serverRoot}/addressbook/{apiVersion}/{userId}/contacts/{contactId}

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user who wants to delete a contact

**contactId:** Identifier (sip or tel) identifying the contact to be deleted

**Response:**

204 code to successful calls or a 4xx code to unsuccessful calls.

### 4.2.10. Retrieve all attributes of a contact

Retrieve all attributes of a contact belonging to a user. The names of the attributes that will be return are display-name and capabilities.

HTTP GET http://{serverRoot}/addressbook/{apiVersion}/{userId}/contacts/{contactId}/attributes

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user who wants to retrieve the attributes of a contact

**contactId:** Identifier (sip or tel) identifying the contact

**Response:**

200 with an AttributeList data structure in the body or a 4xx code if an error happened.

### 4.2.11. Retrieve an attribute of a contact

Retrieve an attribute value of a contact of a user.

HTTP GET http://{serverRoot}/addressbook/{apiVersion}/{userId}/contacts/{contactId}/attributes/[ResourceRelPath]

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user who wants to retrieve an attribute of a contact

**contactId:** Identifier (sip or tel) identifying the contact

**ResourceRelPath:** name of the attribute (display-name or capabilities)

**Response:**

200 with an Attribute data structure in the body or a 4xx code if an error happened.

### 4.2.12. Create/Update an attribute for a contact

This operation is used for creation or update of an attribute of a contact. Only the display-name attribute can be update.

HTTP PUT http://{serverRoot}/addressbook/{apiVersion}/{userId}/contacts/{contactId}/attributes/[ResourceRelPath]

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user who wants to create/update an attribute for a contact

**contactId:** Identifier (sip or tel) identifying the contact

**ResourceRelPath:** name of the attribute (display-name)

**Response:**

200 with an Attribute data structure in the body or a 4xx code if an error happened.

### 4.2.13. Delete an attribute of a contact

This operation is used for creation or update of an attribute of a contact. Only the display-name attribute can be deleted (it sets display-name to the uri of the contact).

HTTP DELETE http://{serverRoot}/addressbook/{apiVersion}/{userId}/contacts/{contactId}/attributes/[ResourceRelPath]

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user who wants to delete an attribute of a contact

**contactId:** Identifier (sip or tehl) identifying the contact

**ResourceRelPath:** name of the attribute (display-name)

**Response:**

204 code to successful calls or a 4xx code to unsuccessful calls.

## 4.3. Chat

### 4.3.1. Create a new subscription to chat notifications

This operation is used to create a subscription in order to receive chat and group chat notifications.

HTTP POST http://{serverRoot}/chat/{apiVersion}/{userId}/subscriptions

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user

**Body:**

ChatNotificationSubscription data structure

**Response:**

200 with a modified ChatNotificationSubscription data structure in the body or a 4xx code if an error happened.

### 4.3.2. Delete a subscription

This operation is used to delete a subscription to chat and group chat notifications.

HTTP DELETE http://{serverRoot}/chat/{apiVersion}/{userId}/subscriptions/{subscriptionId}

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user

**subscriptionId:** Identifier of the subscription that will be delete

**Response:**

204 code to successful calls or a 4xx code to unsuccessful calls.

### 4.3.3. Retrieve information about individual subscription

This operation is used to retrieve information about a subscription for chat and group chat notifications.

HTTP GET http://{serverRoot}/chat/{apiVersion}/{userId}/subscriptions/{subscriptionId}

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user

**subscriptionId:** Identifier of the subscription

**Response:**

200 with a ChatNotificationSubscription data structure in the body or a 4xx code if an error happened.

### 4.3.4. Retrieve information about all active subscriptions

This operation is used to retrieve information about all subscriptions for this resource (only one subscription for this resource can be created in this API version).

HTTP GET http://{serverRoot}/chat/{apiVersion}/{userId}/subscriptions

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user

**Response:**

200 with an ChatSubscriptionList data structure in the body or a 4xx code if an error happened.

### 4.3.5. Create chat

To send messages with the option of session self-management (this step is not necessary for adhoc 1-1 chats), we have several methods. First one is the createChat that creates the chat session.

HTTP POST http://{serverRoot}/chat/{apiVersion}/{userId}/oneToOne/{otherUserId}

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user who is asking the messages he has received.

**otherUserId:** Identifies the destination URI of the user who is receiving the messages.

**Body:**

ChatSessionInformation data structure.

**Response:**

201 with a modified ChatSessionInformation data structure in the body or a 4xx code if an error happened.

### 4.3.6. Send IM Message

Once the chat session is opened, we will use this method to continue sending IM messages.

HTTP POST http://{serverRoot}/chat/{apiVersion}/{userId}/oneToOne/{otherUserId}/{sessionId}/messages

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user who is asking the messages he has received.

**otherUserId:** Identifies the destination URI of the user who is receiving the messages.

**sessionId:** The chat sessionId in which we want to send the messages or 'adhoc' for automatic management of the session.

**Body:**

ChatMessage data structure.

**Response:**

201 with a ResourceReference data structure or a 4xx code if an error happened.

### 4.3.7. Close chat

Terminate the selected chat session. SIP BYE request is sent to remote party.

HTTP DELETE http://{serverRoot}/chat/{apiVersion}/{userId}/oneToOne/{otherUserId}/{sessionId}

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user who wants to close a chat

**otherUserId:** Identifier of the other user in the chat session.

**sessionId:** The chat sessionId which we want to close.

**Response:**

204 code to successful calls or a 4xx code to unsuccessful calls.

### 4.3.8. Send is composing

Send an “is composing” message to the session you have opened. This allows a user in a chat session to see when another user is typing a new message/reaction.

HTTP POST http://{serverRoot}/chat/{apiVersion}/{userId}/oneToOne/{otherUserId}/{sessionId}/messages

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user who is asking the messages he has received.

**otherUserId:** Identifier the destination URI of the user who is receiving the messages.

**sessionId:** The chat sessionId in which we want to send the messages or 'adhoc' for automatic management of the session.

**Body:**

IsComposing data structure.

**Response:**

201 with a ResourceReference data structure in the body or a 4xx code if an error happened.

### 4.3.9. Send display

This method is used to send display notifications. It is needed to configure the user for NOT to send the notifications.

HTTP PUT http://{serverRoot}/chat/{apiVersion}/{userId}/oneToOne/{otherUserId}/{sessionId}/messages/{messageId}/status

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user who is asking the messages he has received.

**OtherUserId:** Identifier of the other user in the chat session.

**sessionId:** The chat session in which the message was sent.

**messageId:** The messageId which we want to report the new status.

**Body:**

MessageStatusReport data structure.

**Response:**

204 code to successful calls or a 4xx code to unsuccessful calls.

### 4.3.10. Accept IM session

This method is used to accept IM invitations. To accept or reject chat sessions (it is independent of the session self-management but the user has to be configured to NOT auto-accept chat sessions)

HTTP PUT http://{serverRoot}/chat/{apiVersion}/{userId}/oneToOne/{otherUserId}/{sessionId}/status

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user who received the invitation.

**otherUserId:** Identifier of the other user in the chat session.

**sessionId:** The chat sessionId which we want to accept.

**Body:**

ParticipantSessionStatus data structure.

**Response:**

204 code to successful calls or a 4xx code to unsuccessful calls.

### 4.3.11. Decline IM session

This method is used to decline IM invitations. To accept or reject chat sessions (it is independent of the session self-management but the user has to be configured to NOT auto-accept chat sessions)

HTTP DELETE http://{serverRoot}/chat/{apiVersion}/{userId}/oneToOne/{otherUserId}/{sessionId}

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user who wants to close a chat

**otherUserId:** Identifier of the other user in the chat session.

**sessionId:** The chat sessionId which we want to decline.

**Response:**

204 code to successful calls or a 4xx code to unsuccessful calls.

### 4.3.12. Read 1-1 chat session information

This operation is used to retrieve chat session information.

HTTP GET http://{serverRoot}/chat/{apiVersion}/{userId}/oneToOne/{otherUserId}/{sessionId}

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user who wants to read the chat session information.

**otherUserId:** Identifier of the other user in the chat session.

**sessionId:** The chat sessionId which we want to read information.

**Response:**

201 code with a ChatSessionInformation in the body or a 4xx code if an error happened.

### 4.3.13. Create group chat

Create a group chat with the contacts selected. Once the group chat is established we will receive a groupChatStablished notification (see Error: Reference source not found). During a group chat, if any contact leaves the conference or join it, we will receive a groupChatUpdated notification.

The result of this operation contains a conference sessionId which we have to use to send messages to the people invited to the group chat.

HTTP POST http://{serverRoot}/chat/{apiVersion}/{userId}/group

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user who wants to create the group chat

**Body:**

GroupChatSessionInformation data structure

**Response:**

201 with a ResourceReference data structure in the body or a 4xx code if an error happened.

### 4.3.14. Add contacts to a group chat

Add contacts to a group chat already created. Once the request is sent, we will receive a GroupChatUpdated notification containing all the contact uris that joined the group.

HTTP POST http://{serverRoot}/chat/{apiVersion}/{userId}/group/{sessionId}/participants

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user who wants to add contacts to the group chat.

**sessionId:** The conference sessionId in which we want to add more participants.

**Body:**

ParticipantList (more than one participant) or ParticipantInformation (one participant) data structures.

**Response:**

201 with a ResourceReference data structure in the body or a 4xx code if an error happened.

### 4.3.15. Send message to a group chat

Sending messages inside a group chat is almost like sending an IM message in a 1 to 1 session once the session is created (see Error: Reference source not found). The difference is that we have to create the group chat before sending any message; all the messages must be sent with the conference sessionId. Unlike the sendIMMessage method, the destination contactUri is not needed because the message will be sent to all the members of the conference.

HTTP POST http://{serverRoot}/chat/{apiVersion}/{userId}/group/{sessionId}/messages

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user who wants to send a message

**sessionId:** the conference sessionId in which we want to send a message to the contacts.

**Body:**

ChatMessage data structure

**Response:**

201 with a ResourceReference data structure in the body or a 4xx code if an error happened.

### 4.3.16. Exit from a group chat

Leave a group chat session

. HTTP DELETE http://{serverRoot}/chat/{apiVersion}/{userId}/group/{sessionId}/participants/{participantId}

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user who wants to close the group chat.

**sessionId:** the conference sessionId in which we want to close the group chat.

**participantId:** Identifier of the user in the group chat session.

**Response:**

204 code to successful calls or a 4xx code to unsuccessful calls.

### 4.3.17. Accept group chat invitation

It accepts a group chat invitation. To accept or reject a group chat invitation, the user has to be configured to NOT auto accept group chat)

HTTP PUT http://{serverRoot}/chat/{apiVersion}/{userId}/group/{sessionId}/participants/{participantId}/status

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user who wants to accept the group chat.

**sessionId:** The conference sessionId in which we want to accept the group chat.

**participantId:** Identifier of the user in the group chat session.

**Body:**

ParticipantSessionStatus data structure.

**Response:**

204 code to successful calls or a 4xx code to unsuccessful calls.

### 4.3.18. Decline group chat invitation

It declines a group chat invitation. To accept or reject a group chat invitation, the user has to be configured to NOT auto accept group chat)

HTTP DELETE http://{serverRoot}/chat/{apiVersion}/{userId}/group/{sessionId}/participants/{participantId}

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user who wants to close the group chat.

**sessionId:** the conference sessionId in which we want to close the group chat.

**participantId:** Identifier of the user in the group chat session.

**Response:**

204 code to successful calls or a 4xx code to unsuccessful calls.

### 4.3.19. Retrieve group chat session information

This operation is used to retrieve chat session information.

HTTP GET http://{serverRoot}/chat/{apiVersion}/{userId}/group/{sessionId}

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user who wants to retrieve group chat session information.

**sessionId:** the conference sessionId.

**Response:**

201 with a GroupChatSessionInformation data structure in the body or a 4xx code if an error happened.

### 4.3.20. Read the list of group chat participants

This operation is used to retrieve the list of Participants in a group chat session.

HTTP GET http://{serverRoot}/chat/{apiVersion}/{userId}/group/{sessionId}/participants

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user who wants to read the participants list.

**sessionId:** the conference sessionId.

**Response:**

201 with a ParticipantList data structure in the body or a 4xx code if an error happened.

#### 4.3.21. Read information about an individual group chat participant

This operation is used to retrieve information about an individual group chat Participant.

HTTP GET http://{serverRoot}/chat/{apiVersion}/{userId}/group/{sessionId}/participants/{participantId}

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user who wants to retrieve information about a participant in the group chat session.

**sessionId:** the conference sessionId.

**participantId:** Identifier of the user in the group chat session.

**Response:**

201 with a ParticipantInformation data structure in the body or a 4xx code if an error happened.

## 4.4. File Transfer

### 4.4.1. Create a new subscription to file transfer notifications

This operation is used to create a subscription in order to receive file transfer notifications.

HTTP POST http://{serverRoot}/filetransfer/{apiVersion}/{userId}/subscriptions

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user

**Body:**

FileTransferSubscription data structure

**Response:**

200 with a modified FileTransferSubscription data structure in the body or a 4xx code if an error happened.

### 4.4.2. Delete a subscription

This operation is used to delete a subscription to file transfer notifications.

HTTP DELETE http://{serverRoot}/filetransfer/{apiVersion}/{userId}/subscriptions/{subscriptionId}

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user

**subscriptionId:** Identifier of the subscription that will be delete

**Response:**

204 code to successful calls or a 4xx code to unsuccessful calls.

### 4.4.3. Retrieve information about individual subscription

This operation is used to retrieve information about a subscription for file transfer notifications.

HTTP GET http://{serverRoot}/filetransfer/{apiVersion}/{userId}/subscriptions/{subscriptionId}

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user

**subscriptionId:** Identifier of the subscription

**Response:**

200 with a FileTransferSubscription data structure in the body or a 4xx code if an error happened.

### 4.4.4. Retrieve information about all active subscriptions

This operation is used to retrieve information about all subscriptions for this resource (only one subscription for this resource can be created in this API version).

HTTP GET http://{serverRoot}/filetransfer/{apiVersion}/{userId}/subscriptions

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user

**Response:**

200 with an FileTransferSubscriptionList data structure in the body or a 4xx code if an error happened.

### 4.4.5. Send file with URL pointer

It sends an url located file to a contact.

HTTP POST http://{serverRoot}/filetransfer/{apiVersion}/{userId}/sessions

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user who wants to send a file

**Body:**

FileTransferSessionInformation data structure

**Response:**

201 with a ResourceReference data structure in the body or a 4xx code if an error happened.

### 4.4.6. Send file data

It sends a file including the file content in the request.

HTTP POST http://{serverRoot}/filetransfer/{apiVersion}/{userId}/sessions

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user who wants to send a file

**Multipart Body:**

*Request Body parameters:*

String body (name root-fields):

FileTransferSessionInformation data structure

File body (name attachments):

File data

**Response:**

201 with a ResourceReference data structure in the body or a 4xx code if an error happened.

### 4.4.7. Accept file transfer

It accepts a file transfer. To accept or reject file transfer, the user has to be configured to NOT auto accept file sessions)

HTTP POST http://{serverRoot}/filetransfer/{apiVersion}/{userId}/sessions/{sessionId}/status

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user who are receiving the file.

**sessionId:** The file transfer sessionId in which we want to accept the invitation.

**Body:**

ReceiverSessionsStatus data structure.

**Response:**

204 code to successful calls or a 4xx code to unsuccessful calls.

### 4.4.8. Decline file transfer

It declines a file transfer. To accept or reject file transfer, the user has to be configured to NOT auto accept file sessions)

HTTP DELETE http://{serverRoot}/filetransfer/{apiVersion}/{userId}/sessions/{sessionId}

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user who wants to decline the file transfer.

**SessionId:** the file transfer sessionId in which we want to decline the invitation.

**Response:**

204 code to successful calls or a 4xx code to unsuccessful calls.

### 4.4.9. Retrieve file transfer session information

This operation is used to retrieve file transfer session information.

HTTP GET http://{serverRoot}/filetransfer/{apiVersion}/{userId}/sessions/{sessionId}

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user who wants to retrieve file transfer session information.

**SessionId:** the file transfer sessionId.

**Response:**

200 with a FileTransferSessionInformation data structure in the body or a 4xx code if an error happened.

## 4.5. Capabilities

### 4.5.1. Retrieve own capabilities

This operation is used to retrieve our capabilities.

HTTP GET http://{serverRoot}/capabilities/{apiVersion}/{userId}

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user

**Response:**

200 with a Capabilities data structure in the body or a 4xx code if an error happened.

### 4.5.2. Set own capabilities

This operation is used to set the capabilities that the other users receive when they do an OPTIONS request.

HTTP PUT http://{serverRoot}/capabilities/{apiVersion}/{userId}

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user

**Body:**

Capabilities data structure.

**Response:**

204 code to successful calls or a 4xx code to unsuccessful calls.

## 4.6. Notification Channel

### 4.6.1. Create a new notification channel

This operation is used to create a new channel that can be used to receive notifications from different resources. Once the channel is created, you can subscribe with it to the chat notifications, file transfer notifications... Only one notification channel can be created in this API version.

HTTP POST http://{serverRoot}/notificationchannel/{apiVersion}/{userId}/channels

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user

**Body:**

NotificationChannel

data structure.

**Response:**

201 with a modified NotificationChannel data structure in the body or a 4xx code if an error happened.

### 4.6.2. Delete a notification channel

This operation is used to delete a notification channel.

HTTP DELETE http://{serverRoot}/notificationchannel/{apiVersion}/{userId}/channels/{channelId}

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user

**channelId:** Identifier of the channel to delete

**Response:**

204 code to successful calls or a 4xx code to unsuccessful calls.

### 4.6.3. Retrieve an individual notification channel

This operation is used to retrieve information about a notification channel.

HTTP GET http://{serverRoot}/notificationchannel/{apiVersion}/{userId}/channels/{channelId}

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user

**channelId:** Identifier of the channel

**Response:**

200 with a NotificationChannel data structure in the body or a 4xx code if an error happened.

### 4.6.4. Retrieve a list of notification channels

This operation is used to retrieve all the active notification channels of a specific user (only one notification channel can be created in this API version).

HTTP GET http://{serverRoot}/notificationchannel/{apiVersion}/{userId}/channels

**Parameters:**

**apiVersion:** 0.1

**userId:** Identifier of the user

**Response:**

200 with a NotificationChannelList data structure in the body or a 4xx code if an error happened.

# 5. Data structures

In this section we will show the data structures used in the OMA API calls

## 5.1. Common data structures

### 5.1.1. Type: ResourceReference

| Element | Type | Optional | Description |
|---|---|---|---|
| resourceURL | anyUri | Yes | Self-referring URL. The resourceURL SHALL NOT be included in the requests by the client. The resourceURL MUST be included in responses. |

### 5.1.2. Type: Link

| Element | Type | Optional | Description |
|---|---|---|---|
| rel | String | No | Describe the relationship between the URI and the resource. |
| href | anyUri | No | URI |

### 5.1.3. Type: RequestError

| Element | Type | Optional | Description |
|---|---|---|---|
| link | Link [0..unbounded] | Yes | Link to elements external to the resource. |
| serviceException | ServiceException | No | Exception details. |
| policyException | PolicyException | Yes | Exception details. |

### 5.1.4. Type: ServiceException

| Element | Type | Optional | Description |
|---|---|---|---|
| messageId | String | No | Message identifier, with prefix SVC. |

| | | | |
|---|---|---|---|
| text | String | No | Message text, with replacement variables marked with %n, where n is an index into the variables list. |
| variables | String [0..unbounded] | Yes | Variables to substitute into Text string. |

### 5.1.5. Type: PolicyException

| Element | Type | Optional | Description |
|---|---|---|---|
| messageId | String | No | Message identifier, with prefix POL. |
| text | String | No | Message text, with replacement variables marked with %n, where n is an index into the variables list. |
| variables | String [0..unbounded] | Yes | Variables to substitute into Text string. |

### 5.1.6. Type: CallbackReference

| Element | Type | Optional | Description |
|---|---|---|---|
| notifyURL | anyUri | No | Notify callback URL. |
| callbackData | String | Yes | Data that the application can set when subscribe to notifications. |
| notificationFormat | NotificationFormat | Yes | Default: JSON<br><br>Application can specify format of the resource representation in notification. The choice is between {XML, JSON}. At this API version only JSON is valid. |

### 5.1.7. Type: ServiceError

This type represents information about the cancellation of a subscription.

| Element | Type | Optional | Description |
|---|---|---|---|
| messageId | String | No | Message identifier, with prefix SVC or POL. |
| text | common: ServiceError | No | Message text, with replacement variables marked with %n, where n is an index into the variables list. |
| variables | String [0..unbounded] | Yes | Variables to substitute into Text string. |

### 5.1.8. Enumeration: NotificationFormat

This enumeration define the different formats to receive a notification.

| Enumeration | Description |
|---|---|

| | |
|---|---|
| XML | Notification about new inbound message would use XML format in the POST request. |
| JSON | Notification about new inbound message would use JSON format in the POST request. |

## 5.2. Session

### 5.2.1. Type: SessionSubscription

This type is used to create a new subscription in order to receive session notifications.

| Element | Type | Optional | Description |
|---|---|---|---|
| callbackReference | Common: CallbackReference | No | Notification URL where the notifications will be send and optional callbackData. |
| clientCorrelator | String | Yes | A correlator that the client can use to tag this particular resource. |
| duration | int | Yes | Subscription duration in second. If not specified the duration of the subscription or is 0, the server will consider this subscription with a infinite duration. |
| resourceURL | anyUri | Yes | Self-referring URL. The resourceURL SHALL NOT be included in the requests by the client. The resourceURL MUST be included in responses. |

### 5.2.2. Type: SessionSubscriptionList

This type represents all the active subscriptions to session notifications.

| Element | Type | Optional | Description |
|---|---|---|---|
| sessionSubscription | sessionSubscription [0..unbounded] | Yes | Array of chat notification subscriptions |
| resourceURL | anyUri | No | Self-referring URL. The resourceURL SHALL NOT be included in the requests by the client. The resourceURL MUST be included in responses. |

### 5.2.3. Type: SubscriptionCancellationNotification

This type represent the information about a notification received when a subscription expires.

| Element | Type | Optional | Description |
|---|---|---|---|

| | | | |
|---|---|---|---|
| callbackData | String | Yes | CallbackData passed by the application in the subscription request. |
| reason | common: ServiceError | Yes | Reason of the cancellation of the subscription. |
| link | common: Link [1..unbounded] | No | There MUST be a link to the subscription that is cancelled. |

### 5.2.4. Type: SessionEventNotification

This type represents the information of a notification received when a user tries register or unregister.

| Element | Type | Optional | Description |
|---|---|---|---|
| callbackData | String | Yes | CallbackData passed by the application in the subscription request. |
| link | common: Link [1..unbounded] | Yes | Link to other resources that are in relationship with this resource. |
| eventType | SessionEvent | No | Type of event |

### 5.2.5. Enumeration: SessionEvent

| Enumeration | Description |
|---|---|
| registerSuccess | The register was successfully |
| registerError | The register was unsuccessfully |
| unregisterSuccess | The unregister was successfully |
| unregisterError | The unregister was unsuccessfully |

## 5.3. Contacts

### 5.3.1. Type: AbChangesSubscription

This type is used to create a new subscription in order to receive contacts notifications.

| Element | Type | Optional | Description |
|---|---|---|---|
| callbackReference | Common: CallbackReference | No | Notification URL where the notifications will be send and optional callbackData. |
| clientCorrelator | String | Yes | A correlator that the client can use to tag this particular resource. |

| | | | |
|---|---|---|---|
| applicationTag | String | Yes | A tag that the client may use to tag this particular resource. |
| duration | int | Yes | Subscription duration in second. If not specified the duration of the subscription or is 0, the server will consider this subscription with a infinite duration. |
| resourceURL | anyUri | Yes | Self-referring URL. The resourceURL SHALL NOT be included in the requests by the client. The resourceURL MUST be included in responses. |

### 5.3.2. Type: AbChangesSubscriptionCollection

This type represents all the active subscriptions to session notifications.

| Element | Type | Optional | Description |
|---|---|---|---|
| abChangesSubscription | AbChangesSubscription [0..unbounded] | Yes | Contains a list of subscriptions to addressbook notifications. |
| resourceURL | anyUri | No | Self-referring URL. The resourceURL SHALL NOT be included in the requests by the client. The resourceURL MUST be included in responses. |

### 5.3.3. Type: ContactCollection

This type represents the list of all individual contact for a specific user.

| Element | Type | Optional | Description |
|---|---|---|---|
| contact | Contact [0..unbounded] | Yes | Contains a list of user's contacts. |
| resourceURL | anyUri | Yes | Self-referring URL. The resourceURL SHALL NOT be included in the requests by the client. The resourceURL MUST be included in responses. |

### 5.3.4. Type: Contact

This type represents the information of a contact.

| Element | Type | Optional | Description |
|---|---|---|---|
| contactId | anyUri | No | Contains de URI of the contact. If contactId is also part of the request URL, the two MUST have the same |

| | | | |
|---|---|---|---|
| | | | value. |
| sharedIdentity | SharedIdentity | Yes | Contains a list of publicly known identities of the contact. |
| attributeList | AttributeList | Yes | Contains a list of attributes (e.g. “display-name”, “capabilities”...) related to a contact. |
| resourceURL | anyUri | Yes | Self-referring URL. The resourceURL SHALL NOT be included in the requests by the client. The resourceURL MUST be included in responses. |
| link | common:Link [0..unbounded] | Yes | MAY contain references to members (rel=”Member”) in type List associated with the contact. |

### 5.3.5. Type: SharedIdentity

List of public identities.

| Element | Type | Optional | Description |
|---|---|---|---|
| sharedId | anyUri [0..unbounded] | Yes | Contains a list of absolute URIs representing contact identities or list identities known to other users. |

### 5.3.6. Type: AttributeList

List of attributes of a contact.

| Element | Type | Optional | Description |
|---|---|---|---|
| attribute | Attribute [0..unbounded] | Yes | Contains a list of attributes related to a contact. |
| resourceURL | anyUri | Yes | Self-referring URL. The resourceURL SHALL NOT be included in the requests by the client. The resourceURL MUST be included in responses. |

### 5.3.7. Type: Attribute

Information about an attribute. There are two valid attributes:

- display-name: name to show for a contact.
- capabilities: capabilities of the user.

| Element | Type | Optional | Description |
|---|---|---|---|
| name | String | No | Name of the attribute. |

| | | | |
|---|---|---|---|
| value | String | Yes | Value of the attribute. |
| objectValue | base64Binary | Yes | Contains a base64Binary object. |

### 5.3.8. Type: SubscriptionCancellationNotification

This type represent the information about a notification received when a subscription expires.

| Element | Type | Optional | Description |
|---|---|---|---|
| callbackData | String | Yes | CallbackData passed by the application in the subscription request. |
| reason | common: ServiceError | Yes | Reason of the cancellation of the subscription. |
| link | common: Link [1..unbounded] | No | There MUST be a link to the subscription that is cancelled. |

### 5.3.9. Type: ContactEventNotification

This type represents the information received for a change in a contact.

| Element | Type | Optional | Description |
|---|---|---|---|
| callbackData | String | Yes | CallbackData passed by the application in the subscription request. |
| link | common: Link [1..unbounded] | Yes | Link to other resources that are in relationship with this resource.<br>The server MUST include links as defined in section 6.1.<br>Further, the server SHOULD include a link to the subscription. |
| contactAddress | anyUri | No | Contact who produced the event. |
| displayName | String | Yes | Display name of that contact. |
| capabilities | String | Yes | Capabilities of that contact. |
| event | ContactEvent | No | Type of event |

### 5.3.10. Enumeration: ContactEvent

| Enumeration | Description |
|---|---|
| ContactAdded | New contact. |

| | |
|---|---|
| ContactChanged | The contact changed his information (capabilities...). |
| ContactRemoved | Now, the contact is not in your contact list. |

## 5.4. Chat

### 5.4.1. Type: ChatNotificationSubscription

This type represent a subscription to chat and group chat events or notifications.

| Element | Type | Optional | Description |
|---|---|---|---|
| callbackReference | Common: CallbackReference | No | Notification URL where the notifications will be send and optional callbackData. |
| clientCorrelator | String | Yes | A correlator that the client can use to tag this particular resource. |
| confirmedChatSupported | boolean | Yes | Default: false<br><br>If is true, the user will manage the sessions for 1-1 chats. |
| adhocChatSupported | boolean | Yes | Default: true<br><br>If is true, the server will manage the session instead the user. |
| duration | int | Yes | Subscription duration in second. If not specified the duration of the subscription or is 0, the server will consider this subscription with a infinite duration. |
| resourceURL | anyUri | Yes | Self-referring URL. The resourceURL SHALL NOT be included in the requests by the client. The resourceURL MUST be included in responses. |

### 5.4.2. Type: ChatSubscriptionList

This type represents all the active subscriptions to chat and group chat notifications.

| Element | Type | Optional | Description |
|---|---|---|---|
| chatNotificationSubscription | ChatNotification Subscription [0..unbounded] | Yes | Array of chat notification subscriptions |
| resourceURL | anyUri | No | Self-referring URL. The resourceURL |

| | | | |
|---|---|---|---|
| | | | SHALL NOT be included in the requests by the client. The resourceURL MUST be included in responses. |

### 5.4.3. Type: ChatSessionInformation

This type represents information about a 1-1 chat session.

| Element | Type | Optional | Description |
|---|---|---|---|
| subject | String | No | Initial message of the chat session. |
| originatorAddress | anyUri | No | Address of the originator. |
| originatorName | String | Yes | Name of the originator. |
| tParticipantAddress | anyUri | No | Address of the terminating participant. |
| tParticipantName | String | Yes | Name of the terminating participant. |
| status | ParticipantStatus | Yes | Connection status of the participant. Set by the server, SHALL NOT be present in the request. |
| resourceURL | anyUri | Yes | Self-referring URL. The resourceURL SHALL NOT be included in the requests by the client. The resourceURL MUST be included in responses. |

### 5.4.4. Type: ChatMessage

This type represent information about a chat message.

| Element | Type | Optional | Description |
|---|---|---|---|
| text | String | No | Text content of a chat message. |
| reportRequest | MessageStatus [0..unbounded] | Yes | List of status events to report. This element is not relevant in group chats. |
| resourceURL | anyUri | Yes | Self-referring URL. The resourceURL SHALL NOT be included in the requests by the client. The resourceURL MUST be included in responses. |

### 5.4.5. Type: IsComposing

This type represents information to send an is composing event to the other participant in an 1-1 chat.

| Element | Type | Optional | Description |
|---|---|---|---|
| state | String | No | Sender state, as defined in [RFC3994]. One of “idle”, “active”. |

| | | | |
|---|---|---|---|
| lastActive | dateTime | Yes | Time of the last activity, as defined in [RFC3994] |
| contentType | String | Yes | Time of message being created, as defined in [RFC3994]. |
| refresh | positiveInteger | Yes | Time interval in seconds after which the Receiver can expect an update from the Sender, as defined in [RFC3994]. |

### 5.4.6. Type: MessageStatusReport

This type represent the status of a message. This type will be send when the other participant want to receive a report about “Displayed” message status.

| Element | Type | Optional | Description |
|---|---|---|---|
| status | MessageStatus | No | Indicate the status of the message. |

### 5.4.7. Type: GroupChatSessionInformation

This type represents information about a group chat session.

| Element | Type | Optional | Description |
|---|---|---|---|
| subject | String | No | Initial message of the chat session. |
| participant | ParticipantInformation [0..unbounded] | No | The participant(s) connected or invited to this chat session. |
| clientCorrelator | String | Yes | A correlator that the client can use to tag this particular resource. |
| resourceURL | anyUri | Yes | Self referring URL. The resourceURL SHALL NOT be included in the requests by the client. The resourceURL MUST be included in responses. |

### 5.4.8. Type: ParticipantInformation

This type represents information about a participant in a group chat.

| Element | Type | Optional | Description |
|---|---|---|---|
| address | anyUri | No | The address of the participant. |
| name | String | Yes | The name of the participant. |
| isOriginator | boolean | Yes | If the participant is the originator of the session, this element MUST be set to “true”. In other cases, it MUST be set to “false” (default value) or ignore it. |

| | | | |
|---|---|---|---|
| status | ParticipantStatus | Yes | Connection status of the participant. Set by the server. SHALL NOT be present in requests. |
| clientCorrelator | String | Yes | A correlator that the client can use to tag this particular resource. |
| resourceURL | anyUri | Yes | Self referring URL. The resourceURL SHALL NOT be included in the requests by the client. The resourceURL MUST be included in responses. |

### 5.4.9. Type: ParticipantList

This type represents a list of participants in a group chat.

| Element | Type | Optional | Description |
|---|---|---|---|
| participant | ParticipantInformation [1..unbounded] | No | List of chat participants. |
| resourceURL | anyUri | Yes | Self-referring URL. The resourceURL SHALL NOT be included in the requests by the client. The resourceURL MUST be included in responses. |

### 5.4.10. Type: ParticipantSessionStatus

This type represents the status of a participant in a group chat.

| Element | Type | Optional | Description |
|---|---|---|---|
| status | ParticipantStatus | No | Status of the participant. To indicate that the client accepts the invitation, this element MUST be set to “Connected”. |

### 5.4.11. Type: ParticipantStatusEntry

This type represents information about the status of a participant in a group chat session.

| Element | Type | Optional | Description |
|---|---|---|---|
| address | anyUri | No | Address of the participant. |
| name | String | Yes | Name of the participant. |
| status | ParticipantStatus | Yes | Connection status of the participant. |
| yourown | boolean | Yes | If present and set to true, this indicates that the status entry represents the Participant to which this data structure is sent in a message. |

| | | | |
|---|---|---|---|
| link | common: Link [1..unbounded] | No | Links to other resource that are in relationship to this resource. |

### 5.4.12. Type: SubscriptionCancellationNotification

This type represent the information about a notification received when a subscription expires.

| Element | Type | Optional | Description |
|---|---|---|---|
| callbackData | String | Yes | CallbackData passed by the application in the subscription request. |
| reason | common: ServiceError | Yes | Reason of the cancellation of the subscription. |
| link | common: Link [1..unbounded] | No | There MUST be a link to the subscription that is cancelled. |

### 5.4.13. Type: ChatSessionInvitationNotification

This type represents the information about a 1-1 chat session invitation.

| Element | Type | Optional | Description |
|---|---|---|---|
| callbackData | String | Yes | CallbackData passed by the application in the subscription request. |
| link | common: Link [1..unbounded] | Yes | Link to other resources that are in relationship with this resource.<br>The server MUST include links as defined in section 6.2.<br>Further, the server SHOULD include a link to the subscription. |
| subject | String | No | First message of the chat session. |
| originatorAddress | anyUri | No | Address of the originator. |
| originatorName | String | Yes | Name of the originator. |
| tParticipantAddress | anyUri | Yes | Address of the receiver. |
| tParticipantName | String | Yes | Name of the receiver. |
| sessionId | String | No | SessionId of the 1-1 chat session. If the user is configured to automatic management of the session, the value of the sessionId will be 'adhoc'. |

### 5.4.14. Type: ChatEventNotification

This types represents information about a notification produced for a chat event.

| Element | Type | Optional | Description |
|---|---|---|---|
| callbackData | String | Yes | CallbackData passed by the application in the subscription request. |
| link | common: Link [1..unbounded] | Yes | Link to other resources that are in relationship with this resource.<br>The server MUST include links as defined in section 6.3.<br>Further, the server SHOULD include a link to the subscription. |
| eventType | EventType | No | Type of event |
| eventDescription | String | Yes | Description of the event. |
| sessionId | String | No | SessionId of the 1-1 chat session. If the user is configured to automatic management of the session, the value of the sessionId will be 'adhoc'. |

### 5.4.15. Type: MessageNotification

This type represent information about an incoming chat message or an isComposing message.

| Element | Type | Optional | Description |
|---|---|---|---|
| callbackData | String | Yes | CallbackData passed by the application in the subscription request. |
| link | common: Link [1..unbounded] | Yes | Link to other resources that are in relationship with this resource.<br>The server MUST include links as defined in section 6.4.<br>Further, the server SHOULD include a link to the subscription. |
| senderAddress | anyUri | No | Address of the sender. |
| senderName | String | Yes | Name of the sender. |
| chatMessage | ChatMessage | Yes | Message. |
| isComposing | IsComposing | Yes | Iscomposing message |

| | | | |
|---|---|---|---|
| dateTime | dateTime | Yes | Time when the message was sent. |
| sessionId | String | No | SessionId of the 1-1 chat session. If the user is configured to automatic management of the session, the value of the sessionId will be 'adhoc'. |
| messageId | String | No | Identifier of the message. |

### 5.4.16. Type: MessageStatusNotification

This type represents information about the status of a message.

| Element | Type | Optional | Description |
|---|---|---|---|
| callbackData | String | Yes | CallbackData passed by the application in the subscription request. |
| link | common: Link [1..unbounded] | Yes | Link to other resources that are in relationship with this resource.<br><br>The server MUST include links as defined in section 6.5.<br><br>Further, the server SHOULD include a link to the subscription. |
| status | MessageStatus | No | Status of the message. |
| errorCode | String | Yes | Error code, if any. |
| description | String | Yes | Error description, if any. |
| messageId | String | No | Identifier of the message. |

### 5.4.17. Type: GroupSessionInvitationNotification

This type represents information about an incoming invitation for a group chat session.

| Element | Type | Optional | Description |
|---|---|---|---|
| callbackData | String | Yes | CallbackData passed by the application in the subscription request. |
| link | common: Link [1..unbounded] | Yes | Link to other resources that are in relationship with this resource.<br><br>The server MUST include links as defined in section 6.6.<br><br>Further, the server SHOULD include a link to the subscription. |

| | | | |
|---|---|---|---|
| subject | String | No | Initial message. |
| participant | ParticipantInformation [2..unbounded] | Yes | List of participants in the group chat session. |
| sessionId | String | No | Identifier of the session. |

### 5.4.18. Type: ParticipantStatusNotification

This type represents information about the status of participant in a group chat session.

| Element | Type | Optional | Description |
|---|---|---|---|
| callbackData | String | Yes | CallbackData passed by the application in the subscription request. |
| link | common: Link [1..unbounded] | Yes | Link to other resources that are in relationship with this resource.<br>The server MUST include links as defined in section 6.7.<br>Further, the server SHOULD include a link to the subscription. |
| participant | ParticipantStatusEntry [1..unbounded] | No | List of participants in the group chat session. |
| sessionId | String | No | Identifier of the session. |

### 5.4.19. Enumeration: ParticipantStatus

This enumeration defines the values for a participant in a chat.

| Enumeration | Description |
|---|---|
| Invited | Participant was invited to the session. |
| Connected | Participant is connected to the session. |
| Disconnected | Participant is disconnected from the session. |

### 5.4.20. Enumeration: MessageStatus

This enumeration defines the possibles values for the message status.

| Enumeration | Description |
|---|---|
| Sent | Message was sent. Initial status of a message, not used in PUT request from the client. |

| | |
|---|---|
| Delivered | Message was delivered to the client. Only used in notifications from the server, but not in PUT requests from the client. |
| Displayed | Message was displays by the client. |
| Failed | Message was not delivered to the client. Only used in notifications from the server. |

### 5.4.21. Enumeration: EventType

This enumeration defines the different events in a chat session.

| Enumeration | Description |
|---|---|
| SessionCancelled | The originator has cancelled the chat session during the invite phase. |
| SessionEnded | The chat session has ended. |
| Declined | The participant has declined the chat session. |
| Accepted | The participant has accepted the chat session. |
| Timeout | The participant has not accepted the invitation and the session has timed out. |
| Unreachable | The participant could not be reached or not exists. |

## 5.5. File transfer

### 5.5.1. Type: FileTransferSubscription

This type represent a subscription to file transfer notifications.

| Element | Type | Optional | Description |
|---|---|---|---|
| callbackReference | Common: CallbackReference | No | Notification URL where the notifications will be send and optional callbackData. |
| clientCorrelator | String | Yes | A correlator that the client can use to tag this particular resource. |
| duration | int | Yes | Subscription duration in second. If not specified the duration of the subscription or is 0, the server will consider this subscription with a infinite duration. |
| resourceURL | anyUri | Yes | Self-referring URL. The resourceURL SHALL NOT be included in the requests by the client. The resourceURL MUST be included in |

| | | | |
|---|---|---|---|
| | | | responses. |

### 5.5.2. Type: FileTransferSubscriptionList

This type represents all the active subscriptions file transfer notifications.

| Element | Type | Optional | Description |
|---|---|---|---|
| fileTransferSubscription | FileTransferSubscription [0..unbounded] | Yes | Array of chat notification subscriptions |
| resourceURL | anyUri | No | Self-referring URL. The resourceURL SHALL NOT be included in the requests by the client. The resourceURL MUST be included in responses. |

### 5.5.3. Type: FileTransferSessionInformation

This type represents information about a file transfer session.

| Element | Type | Optional | Description |
|---|---|---|---|
| originatorAddress | anyUri | No | Address of the originator. |
| originatorName | String | Yes | Name of the originator. |
| receiverAddress | anyUri | No | Address of the receiver. |
| receiverName | String | Yes | Name of the receiver. |
| status | ReceiverStatus | Yes | Connection status of the receiver. Set by the server. SHALL NOT be present in the requests. |
| fileInformation | FileInformation | No | Information about the file. |
| clientCorrelator | String | Yes | A correlator that the client can use to tag this particular resource. |
| resourceURL | anyUri | Yes | Self-referring URL. The resourceURL SHALL NOT be included in the requests by the client. The resourceURL MUST be included in responses. |

### 5.5.4. Type: FileInformation

This type represents the information about a file.

| Element | Type | Optional | Description |
|---|---|---|---|

| | | | |
|---|---|---|---|
| fileSelector | FileSelector | No | A tuple of attributes. |
| fileDisposition | FileDisposition | Yes | It is used by the file sender to indicate a preferred disposition of the file. “Render” to indicate that the file should be automatically render or “Attachment” (default value). |
| fileDescription | String | Yes | Short description of the file which could be set by the originator. |
| fileDate | FileDate | Yes | The dates on which the file was created, modified or last read. |
| fileIcon | anyUri | Yes | A small preview icon when the files are images. The value contains a Content-ID URL pointing to an additional body that contains the actual icon in a MIME multipart/related body. |
| fileURL | anyUri | Yes | The URL link to actual file content. If it is present, it indicates that there is not content included in the request operation. If it is not present, it indicates that the content of the file is included in the request body. |
| resourceURL | anyUri | Yes | Self-referring URL. The resourceURL SHALL NOT be included in the requests by the client. The resourceURL MUST be included in responses. |

### 5.5.5.Type: FileSelector

This type represents a list of basic attributes of a file.

| Element | Type | Optional | Description |
|---|---|---|---|
| name | String | No | The name of the file. |
| type | String | No | The MIME type of the file. |
| size | unsignedLong | Yes | The size of the file in octets. |
| hash | HashInformation | Yes | The file hash information. |
| resourceURL | anyUri | Yes | Self-referring URL. The resourceURL SHALL NOT be included in the requests by the client. The resourceURL MUST be included in responses. |

### 5.5.6. Type: FileDate

This type represents the date on wich the file was created, modifies or last read.

| Element | Type | Optional | Description |
|---|---|---|---|
| cDate | dateTime | Yes | The date on which the file was created. |
| mDate | dateTime | Yes | The date on which the file was modified. |
| rDate | dateTime | Yes | The date on which the file was last read. |

### 5.5.7. Type: HashInformation

This type represents the file hash information.

| Element | Type | Optional | Description |
|---|---|---|---|
| algorithm | String | No | The hash algorithm used. |
| value | hexBinary | No | The hash value of the file. |

### 5.5.8. Type: ReceiverSessionStatus

This type represents the status of the receiver in a file transfer session.

| Element | Type | Optional | Description |
|---|---|---|---|
| status | ReceiverStatus | No | Status of the receiver. To indicate that the receiver accepts the session invitation, this element MUST be set to “Connected”. |
| acceptedFile | anyUri | Yes | URL of the accepted file. This element is not necessary, because only one file can be transferred in the same session. |

### 5.5.9. Type: SubscriptionCancellationNotification

This type represent the information about a notification received when a subscription expires.

| Element | Type | Optional | Description |
|---|---|---|---|
| callbackData | String | Yes | CallbackData passed by the application in the subscription request. |
| reason | common: ServiceError | Yes | Reason of the cancellation of the subscription. |
| link | common: Link [1..unbounded] | No | There MUST be a link to the subscription that is cancelled. |

### 5.5.10. Type: FTSessionInvitationNotification

This type represents information about an incoming invitation for a file transfer session.

| Element | Type | Optional | Description |
|---|---|---|---|
| callbackData | String | Yes | CallbackData passed by the application in the subscription request. |
| link | common: Link [1..unbounded] | Yes | Link to other resources that are in relationship with this resource.<br><br>Further, the server SHOULD include a link to the subscription. |
| originatorAddress | anyUri | No | Address of the originator. |
| originatorName | String | Yes | Name of the originator. |
| receiverAddress | anyUri | Yes | Address of the receiver. |
| receiverName | String | Yes | Name of the receiver. |
| fileInformation | FileInformation | No | Information of the file. |
| sessionId | String | No | Identifier of the session. |

### 5.5.11. Type: FileNotification

This type represents information about a delivered file.

| Element | Type | Optional | Description |
|---|---|---|---|
| callbackData | String | Yes | CallbackData passed by the application in the subscription request. |
| link | common: Link [1..unbounded] | Yes | Link to other resources that are in relationship with this resource.<br><br>Further, the server SHOULD include a link to the subscription. |
| fileInformation | FileInformation | No | Information of the file. |
| sessionId | String | No | Identifier of the session. |

### 5.5.12. Type: ReceiverAcceptanceNotification

This type represents information receive when the receiver accepts the file transfer session.

| Element | Type | Optional | Description |
|---|---|---|---|

| | | | |
|---|---|---|---|
| callbackData | String | Yes | CallbackData passed by the application in the subscription request. |
| link | common: Link [1..unbounded] | Yes | Link to other resources that are in relationship with this resource. Further, the server SHOULD include a link to the subscription. |
| receiverAddress | anyUri | No | Address of the receiver. |
| receiverName | String | Yes | Name of the receiver. |
| receiverSessionStatus | ReceiverSessionStatus | No | Connection status of the receiver. |
| sessionId | String | No | Identifier of the session. |

### 5.5.13. Type: FileTransferEventNotification

This typre represents information about events during the file transfer session.

| Element | Type | Optional | Description |
|---|---|---|---|
| callbackData | String | Yes | CallbackData passed by the application in the subscription request. |
| link | common: Link [1..unbounded] | Yes | Link to other resources that are in relationship with this resource. The server MUST include links as defined in section 6.8. Further, the server SHOULD include a link to the subscription. |
| eventType | EventType | No | Type of the event. |
| eventDescription | String | Yes | Description of the event. |
| sessionId | String | No | Identifier of the session. |

### 5.5.14. Enumeration: FileDisposition

This enumeration defines the different dispositions of a file.

| Enumeration | Description |
|---|---|
| Render | Indicates that the file should be automatically rendered. |
| Attachment | Indicates that the file should not be automatically rendered. |

### 5.5.15. Enumeration: ReceiverStatus

This enumeration defines the status of the receiver in a file transfer session.

| Enumeration | Description |
|---|---|
| Invited | User was invited to the session. |
| Connected | User is connected to the session. |
| Disconnected | User is disconnected from the session. |

### 5.5.16. Enumeration: EventType

This enumeration define the different events in a file transfer session.

| Enumeration | Description |
|---|---|
| SessionCancelled | The originator has cancelled the file transfer session during the invite phase. |
| SessionEnded | The file transfer session has ended. |
| Declined | The receiver has declined the file transfer session. |
| Successful | The file was successfully delivered. |
| Failed | The file delivery has failed. |
| Aborted | The file delivery was aborted by the originator. |

## 5.6. Notification Channel

### 5.6.1. Type: NotificationChannel

This type represents information about a notification channel.

| Element | Type | Optional | Description |
|---|---|---|---|
| clientCorrelator | String | Yes | A correlator that the client can use to tag this particular resource. |
| applicationTag | String | Yes | A tag that the client may use to tag this particular resource. |
| channelType | ChannelType | No | Specifies the type on nofitication channel to be used. |
| channelData | ChannelData | Yes | Contains specific information for the notification channel specified in channelType.<br><br>The channelData MUST be included in the |

| | | | |
|---|---|---|---|
| | | | response to the request. |
| channelLifetime | int | Yes | Duration in second. If not specified or is 0, the lifetime, the server will consider this channel with a infinite duration. If this channel expires, the subscriptions using this channel will be cancelled. |
| callbackURL | anyURI | Yes | The callbackURL SHALL NOT be included in POST request. MUST be included in the response. This attribute is used when establishing subscriptions for notification from the different resources (chat, file transfer...) |
| resourceURL | anyUri | Yes | Self-referring URL. The resourceURL SHALL NOT be included in the requests by the client. The resourceURL MUST be included in responses. |

### 5.6.2. Type: NotificationChannelList

This type represents a collection of notification channels of a user (only one notification channel can be created in this API version).

| Element | Type | Optional | Description |
|---|---|---|---|
| notificationChannel | NotificationChannel [0..unbounded] | Yes | Array of chat notification subscriptions |
| resourceURL | anyUri | No | Self-referring URL. The resourceURL SHALL NOT be included in the requests by the client. The resourceURL MUST be included in responses. |

### 5.6.3. Type: ChannelData

This is an abstract data type that contains no elements. The specific channelData for a specific channel type SHALL be derived from this data type.

### 5.6.4. Type: LongPollingData

This type represents a channel data when the mechanism to receive notifications is long polling.

| Element | Type | Optional | Description |
|---|---|---|---|

| | | | |
|---|---|---|---|
| channelURL | anyUri | Yes | ChannelURL SHALL NOT be included in the request, but MUST be included in the response. It contains the URL used to retrieve notifications. |
| maxNotifications | int | Yes | Defines de maximun number of notifications that may be delivered in a notification list. |

### 5.6.5. Enumeration: ChannelType

This enumeration define the types of channels.

| Enumeration | Description |
|---|---|
| LongPolling | Indicates that the mechanism to receive notification through the channel is long polling. |

## 5.7. Capabilities

### 5.7.1. Type: Capabilities

| Element | Type | Optional | Description |
|---|---|---|---|
| address | anyUri | No | The address of the user. |
| imSession | boolean | Yes | True if the user has the instant messaging capability on (default false). |
| fileTransfer | boolean | Yes | True if the user has the file transfer capability on (default false). |
| imageShare | boolean | Yes | True if the user has the image share capability on (default false). |
| videoShare | boolean | Yes | True if the user has the video share capability on (default false). |
| socialPresence | boolean | Yes | True if the user has the social presence capability on (default false). |
| discoveryPresence | boolean | Yes | True if the user has the discovery presence capability on (default false). |

# 6. Notification Links

## 6.1. Client notification about contact events

The server MUST include the following links in the notification.

| EventType | Link rel | Link href Base URL: http://{serverRoot}/addressbook/{apiVersion}/{userId}/contacts |
|---|---|---|
| ContactAdded | ContactInformation | /{contactId} |
| ContactChanged | ContactInformation | /{contactId} |

## 6.2. Client notification about 1-1 chat invitations

- Those links are not relevant in Ad-hoc 1-1 chats and group chats.
- To the other chats session, the following link MUST be included in the notification.

| EventType | Link rel | Link href Base URL: http://{serverRoot}/addressbook/{apiVersion}/{userId}/oneToOne |
|---|---|---|
| n/a | ChatSessionInformation | /{otherUserId}/{sessionId} |

If the application responds with a PUT to this link, the session will be accepted.

If the application responds with a DELETE to this link, the session will be declined.

## 6.3. Client notification about chat session events

- Those links are not relevant in Ad-hoc 1-1 chats.
- Depending the types of the event in a 1-1 chat session, the server MUST include the following links in the notification.

| EventType | Link rel | Link href Base URL: http://{serverRoot}/chat/{apiVersion}/{userId}/oneToOne |
|---|---|---|
| Accepted | ChatSessionInformation | /{otherUserId}/{sessionId} |
| Declined | ChatSessionInformation | /{otherUserId}/{sessionId} |
| SessionCancelled | ChatSessionInformation | /{otherUserId}/{sessionId} |
| SessionEnded | ChatSessionInformation | /{otherUserId}/{sessionId} |

- If the event is in a group chat session, the links to be included are in the next table.

| EventType | Link rel | Link href Base URL: {serverRoot}/chat/{apiVersion}/{userId}/group |
|---|---|---|
| SessionEnded | ChatSessionInformation | /{sessionId} |

## 6.4. Client notification about incoming messages

- In 1-1 chat sessions, the server MUST include the following links in the notification.

| EventType | Link rel | Link href Base URL: http://{serverRoot}/chat/{apiVersion}/{userId}/oneToOne |
|---|---|---|
| n/a | MessageStatusReport | /{otherUserId}/{sessionId}/messages/{messageId}/status |

If the application responds with a PUT to this link, the message will be reported as displayed.

## 6.5. Client notification about message status report

- In 1-1 chat sessions, the server MUST include the following links in the notification.

| EventType | Link rel | Link href Base URL: http://{serverRoot}/chat/{apiVersion}/{userId}/oneToOne |
|---|---|---|
| n/a | ChatMessage | /{otherUserId}/{sessionId}/messages/{messageId} |

## 6.6. Client notification about group chat invitations

- Those links are not relevant in 1-1 chats.
- In group chat sessions, the server MUST include the following links in the notification.

| EventType | Link rel | Link href Base URL: http://{serverRoot}/chat/{apiVersion}/{userId}/group |
|---|---|---|
| n/a | ParticipantInformation | /{sessionId}/participants/{participantId} |
| n/a | ParticipantInformationStatus | /{sessionId}/participants/{participantId}/status |
| n/a | GroupChatSessionInformation | /{sessionId} |

The firs one is used to decline (DELETE) the group chat session invitation and the second one (PUT) is to accept it.

## 6.7. Client notification about participant status in a group chat

- Those links are not relevant in 1-1 chats.
- In group chat sessions, the server MUST include the following links in the notification.

| EventType | Link rel | Link href Base URL: http://{serverRoot}/chat/{apiVersion}/{userId}/group |
|---|---|---|
| n/a | GroupChatSessionInformation | /{sessionId} |

## 6.8. Client notification about file transfer session events

- In file transfer sessions, the server MUST include the following links in the notification.

| EventType | Link rel | Link href<br>Base URL: http://{serverRoot}/filetransfer/{apiVersion}/{userId}/sessions |
|---|---|---|
| n/a | FileTransferSessionInformation | /{sessionId} |

# 7. Examples

On this chapter, we include different examples of services created using the OMA APIs, to help developers to understand how simple is creating them using the API.

## 7.1. Register API

- Resource URL :

  http://{RCSGW-IP}/rcsbox-servlet-oma/register/{apiVersion}/{userId}/sessions

| | |
|---|---|
| HTTP METHOD | POST |
| PARAMETERS IN URL | - apiVersion : API version, 0.1 for the current document<br>- userId : Username of the user to register |
| REQUEST BODY STRUCTURE | ------ |
| RESPONSE BODY STRUCTURE | ------ |
| STATUS CODES | - 204 : Register OK.<br>- 2xx : Message sent with generating a resource in the json response.<br>- 4xx : Register ERROR. Status code contain an error clue. |

**EXAMPLE CODE**

In the example below we have a register request to the Register API Resource.

First of all a senderClient is created. Then the register() method is executed and as we can see the POST http method is prepared. Once the request is done we just have to read the status code and check if the request was success.

```
public class OMA_Register {

        //HttpClient
        protected static final AsyncHttpClient senderclient = createAsyncClient();
        //Resource URI
        private static final String BASE_PATH = "http://80.81.127.142:8280/";

        //Create async client for request
        public static AsyncHttpClient createAsyncClient(){
                AsyncHttpClientConfig.Builder builderSend = new
        AsyncHttpClientConfig.Builder();
                builderSend.setExecutorService(Executors.newCachedThreadPool());
                final ScheduledExecutorService reaperExecSendFT =
        Executors.newSingleThreadScheduledExecutor();
                builderSend.setScheduledExecutorService(reaperExecSendFT);
                AsyncHttpClient asyncHttpClient = new AsyncHttpClient(builderSend.build());
                return asyncHttpClient;
        }

        //Register Request
        public static boolean register( String username){
                try{
                        // -- Sync Request execution --
                        BoundRequestBuilder reqBregister =
        senderclient.preparePost(String.format("%s%s", BASE_PATH, "rcsbox-          servlet-
oma/register/0.1/" + username + "/sessions"));

                        // -- Sync response management --
                        Response response=reqBregister.execute().get();
                        int status=response.getStatusCode();
                        if( (status - 200) < 99 && (status - 200) >= 0 ){//2xx status codes
                                System.out.println("HTTP "+status+" OK. Register ok.");
                                return true;
                        }else{//4xx status codes and others
                                System.err.println(" [HTTP ERROR Registering - "+status+":
                "+response.getStatusText()+"] " + new
        Date(System.currentTimeMillis())+": User " + username);
```

```
                                        return false;

                                }

                        }catch(Exception e){

                                System.err.println("REGISTER ERROR: username="+username+", cause:
                        "+e.getCause());

                                return false;

                        }

                }


                public static void main(String args[]){

                        OMA_Register.register("ExampleUser");

                }

}
```

## 7.2. Unregister API

- Resource URL :

    http://{RCSGW-IP}/rcsbox-servlet-oma/register/{apiVersion}/{userId}/sessions

| | |
|---|---|
| HTTP METHOD | DELETE |
| PARAMETERS IN URL | - apiVersion : API version, 0.1 for the current document<br>- userId : Username of the user to unregister |
| REQUEST BODY STRUCTURE | ------ |
| RESPONSE BODY STRUCTURE | ------ |
| STATUS CODES | - 204 : Unregister OK.<br>- 2xx : Message sent with generating a resource in the json |

| | |
|---|---|
| | response.<br>- 4xx : Unregister ERROR. Status code contain an error clue. |

**EXAMPLE CODE**

```
public class OMA_Unregister {


        protected static final AsyncHttpClient senderclient = createAsyncClient();

        private static final String BASE_PATH = "http://80.81.127.142:8280/";


        //Create async client for request

        public static AsyncHttpClient createAsyncClient(){

                AsyncHttpClientConfig.Builder builderSend = new
                AsyncHttpClientConfig.Builder();

                builderSend.setExecutorService(Executors.newCachedThreadPool());

                final ScheduledExecutorService reaperExecSendFT =
        Executors.newScheduledThreadPool(5);

                builderSend.setScheduledExecutorService(reaperExecSendFT);

                AsyncHttpClient asyncHttpClient = new AsyncHttpClient(builderSend.build());

                return asyncHttpClient;

        }


        //Unregister request

        public static boolean unregister( String username){

                try{

                        // -- Sync Request execution --

                        BoundRequestBuilder reqBunregister =
                senderclient.prepareDelete(String.format("%s%s", BASE_PATH, "rcsbox-
        servlet-oma/register/0.1/" + username + "/sessions"));


                        // -- Sync response management --

                        Response response=reqBunregister.execute().get();

                        int status=response.getStatusCode();

                        if( (status - 200) < 99 && (status - 200) >= 0 ){//2xx status codes

                                System.out.println("HTTP "+status+" OK.Unregister OK.");

                                return true;

                        }else{//4xx status codes and others

                                System.err.println(" [HTTP ERROR RESPONSE UNREGISTERING
                                "+status+": "+response.getStatusText()+"] " + new
                                Date(System.currentTimeMillis())+": User " + username);

                                return false;
```

```
                              }
                    }catch(Exception e){
                              System.err.println("UNREGISTER ERROR: username="+username+", cause:
                    "+e.getCause());
                              return false;
                    }
          }

          public static void main(String args[]){
                    OMA_Unregister.unregister("ExampleUser");
          }
}
```

## 7.3. Get Contact List API

- Resource URL :

    http://{RCSGW-IP}/rcsbox-servlet-oma/addressbook/{apiVersion}/{userId}/contacts

| | |
|---|---|
| HTTP METHOD | GET |
| PARAMETERS IN URL | - apiVersion : API version, 0.1 for the current document<br>- userId :  Username of the user whose will be retrieved |
| REQUEST BODY STRUCTURE | ------ |
| RESPONSE BODY STRUCTURE | JSON<br>**contactCollection**<br>**contact**<br>**0**<br>**attributeList**<br>**attribute**<br>**0**<br>**value**: ExampleContact1<br>**name**: display-name<br>**1**<br>**value**<br>**name**: capabilities |

| | |
|---|---|
| | **resourceURL**: http://80.81.127.142:8280/rcsbox-servlet-oma/addressbook/0.1/loadGeneratorViriato1/contacts/sip:ExampleContact1@solaiemes.com<br>**contactId**: sip:ExampleContact1@solaiemes.com<br>**1**<br>**attributeList**<br>**attribute**<br>**0**<br>**value**: ExampleContact2<br>**name**: display-name<br>**1**<br>**value**<br>**name**: capabilities<br>**resourceURL**: http://80.81.127.142:8280/rcsbox-servlet-oma/addressbook/0.1/loadGeneratorViriato1/contacts/sip:loadGeneratorViriato1@solaiemes.com<br>**contactId**: sip:ExampleContact2@solaiemes.com<br>**resourceURL**: http://80.81.127.142:8280/rcsbox-servlet-oma/addressbook/0.1/ ExampleContact2/contacts |
| STATUS CODES | - 204 : ContactList retrieved OK.<br>- 2xx : Message sent with generating a resource in the json response.<br>- 4xx : ContactList request ERROR. Status code contain an error clue. |

## EXAMPLE CODE

```
public class OMA_GetContactList {

.

.

. // -- HTTP Client creation as in examples before --

.

        public static boolean getContactList( String username){

                try{

                        // -- Sync Request execution --

                        BoundRequestBuilder reqBregister =
                senderclient.prepareGet(String.format("%s%s", BASE_PATH, "rcsbox-
        servlet-oma/addressbook/0.1/" + username + "/contacts"));


                        // -- Sync response management --

                        Response response=reqBregister.execute().get();

                        int status=response.getStatusCode();
                        if( (status - 200) < 99 && (status - 200) >= 0 ){//2xx status codes
                                System.out.println("HTTP "+status+" OK. Contact list 
                        received ok. Response: "+response.getResponseBody());
                                showContactList(response.getResponseBody());
                                return true;
                        }else{//4xx status codes
                                System.err.println(" [HTTP ERROR RETRIEVING CONTACT LIST 
                                "+status+": "+response.getStatusText()+"] " + new 
                                Date(System.currentTimeMillis())+": User " + username);
                                return false;
                        }
                }catch(Exception e){
                        System.err.println("GetContactList ERROR: username="+username+", 
                        cause: "+e.getCause());
                        return false;
                }
        }



        //Search contact list data in json list object and print it.
        public static void showContactList(String jsonResponse){
                //Proccess json response
                JSONObject cometResponse = null;
                try {
                        //Parse to JSONObject
                        cometResponse = (JSONObject) JSONSerializer.toJSON(jsonResponse);

                        //Search data
                        JSONObject contactCollection = 
                        cometResponse.getJSONObject("contactCollection");
                        JSONArray contactList = contactCollection.getJSONArray("contact");
                        System.out.println(contactList.size());
                        for(int i = 0; i < contactList.size(); i++){
                                //Contact Data Object
                                JSONObject contactData = contactList.getJSONObject(i);

                                //Get contactId
                                String contactId = contactData.getString("contactId");
                                String displayName = null;
```

```
                                        //User attributes

                                        JSONObject attributeList =
                              contactData.getJSONObject("attributeList");

                                        JSONArray attributes =
                                        attributeList.getJSONArray("attribute");

                                        for(int h = 0; h < attributes.size() ; h++){//Looking for
                                                                      display-name value

                                                  JSONObject attribute = attributes.getJSONObject(h);

                                                  if( "display-
                                                            name".equalsIgnoreCase(

                                                            attribute.getString("name")) ){

                                                            displayName = attribute.getString("value");

                                                  }

                                        }


                                        //Show contact data

                                        System.out.println(" - User: "+displayName+", uri:
                              "+contactId);

                              }

                    }catch (JSONException e) {

                              e.printStackTrace();

                              System.err.println("ERROR: Invalid json received: " + jsonResponse);

                    }


          }


          public static void main(String args[]){

                    String username = "ExampleUserId";

                    OMA_GetContactList.getContactList(username);


          }
```

## 7.4. Get Contact List API

In this example we send a RCS message between two users. To do it we just have to send a json containing the text message to send to the resource url.

- Resource URL :

  http://{RCSGW-IP} /rcsbox-servlet- oma/chat/{apiVersion}/{userId}/oneToOne/{otherUserId}/{sessionId}/messages

| | |
|---|---|
| HTTP METHOD | POST |
| PARAMETERS IN URL | - apiVersion : API version, 0.1 for the current document<br>- userId : Username of the user who want to send the message<br>- otherUserUri : Tel or sip URI of the destination user<br>- sessionId : Session Id of the user or 'adhoc' if we to autimaticaly manage the sessionId (current case). |
| REQUEST BODY STRUCTURE | JSON<br>{"chatMessage":<br>{"text":"Hello world!"}<br>} |
| RESPONSE BODY STRUCTURE | JSON<br>(With a resource url to get data about the message sent)<br>{"resourceReference":<br>{"resourceURL":"http://192.168.0.48:8080/rcsbox-servlet-oma/chat/0.1/OmaExample1/oneToOne/sip:385914372@solaiemes.com/adhoc/messages/1348219134189-1880165962/status"}<br>} |
| STATUS CODES | - 204 : Message sent OK. Not resource generate in response. Empty response<br>- 2xx : Message sent with generating a resource in the json response.<br>- 4xx : Message sent ERROR. Status code contain an error clue. |

**EXAMPLE CODE**

The ChatMessage object is a java object representation of the request json object needed to sent a message. By the way, the ChatMessage.toString() method return the json text to sent in the request

```
public class OMA_SendIMAutomaticSessionMessage {


        protected static final AsyncHttpClient senderclient = createAsyncClient(); //Asynclient to receive
notifications

        private static final String BASE_PATH = "http://192.168.0.48:8080/";


        //Create async client for request
```

```
public static AsyncHttpClient createAsyncClient(){

            if(senderclient == null){//Dont exists

                        AsyncHttpClientConfig.Builder builderSend = new
                        AsyncHttpClientConfig.Builder();

                        builderSend.setExecutorService(Executors.newCachedThreadPool());

                        final ScheduledExecutorService reaperExecSendFT =
Executors.newScheduledThreadPool(5);

                        builderSend.setScheduledExecutorService(reaperExecSendFT);

                        AsyncHttpClient asyncHttpClient = new
                        AsyncHttpClient(builderSend.build());

                        return asyncHttpClient;

            }else{ //Asynclient already exists

                        return senderclient;

            }

}


public static void sendIMAutomaticMessage( String username, String destUri, String message ){

try{

                        //SEND PARAMS FORMAT

                        BoundRequestBuilder reqBsender =
                        senderclient.preparePost(String.format("%s%s", BASE_PATH,
            "rcsbox-servlet-oma/chat/0.1/" + username + "/oneToOne/" +
destUri + "/adhoc/messages"));

                        //Creating of message object to send

                        ChatMessage cm = new ChatMessage();

                        cm.setText(message);

                        System.out.println("Send message request:\n" + cm.toString());

                        //Get the ChatMessage json string

                        reqBsender.setBody(cm.toString());


                        //Print json to send

                        System.out.println(" - Sending: "+ cm.toString());


                        // -- Async response mamangement --

                        reqBsender.execute(new AsyncCompletionHandler<Response>() {

                                    @Override

                                    public Response onCompleted(Response response) throws
                                                                        Exception {

                                                int status = response.getStatusCode();

                                                if( (status - 200) < 99 && (status - 200) >= 0
                                                            ){//2xx status codes
```

```
                                                    final String jsonResponse = new
                                          String(response.getResponseBodyAsBytes(),
                                "UTF-8");

                                                    System.out.println("HTTP response
                                OK: " +response.getStatusCode());

                                                    System.out.println("response: " +
                                jsonResponse);

                                          }else{

                                                    System.err.println(" HTTP Error
                                response: " +response.getStatusCode());

                                                    System.err.println(" response text: "
                                +response.getStatusText());

                                          }

                                          return null;

                                }

                                @Override

                                public void onThrowable(Throwable t){

                                          System.err.println("ERROR: run - onThrowable: "+
t.getCause());

                                }

                      });


            }catch(Exception e){

                      System.err.println("SENDIMMESSAGE ERROR: username="+username+",
destination uri="+destUri+", message="+message+", cause: "+e.getCause());

            }

  }


  public static void main(String args[]){

            //Message data

            String username = "OmaExample1";

            String destUri = "sip:385914372@solaiemes.com";

            String message = "Hello world!";

            //Execute sendMessage

            OMA_SendIMAutomaticSessionMessage.sendIMAutomaticMessage(username, destUri,
                                                                message);


  }

}
```

## 7.5. Simple echo service example

In this example its created an easy RCS Example Service that listen notification and in case of a MessageNotification will response a echo message to the sender.

The service folow the steps below:

- The main step is create a servlet where the user notifications will be received. We can deploy the servlet in any server and implementing the POST method we will get all the notifications.
- The Solaiemes RCS Solution Gateway need to know where have to send the user notifications. For the reason the user have to subscribe herself to a 'Notification URL' using the subscribing API as in the example below.

In the example, the Servlet classs implement a ServletContextListener that subscribe the user at servlet deployment time. To do that we have the subscriber(<username>) method wich is executed by contextInitialized() method of the ServletContextListener interface. The user registration takes place at the deployment time as the subscription.

- Once the servlet is deployed and the user is subscribed and registered in the RCS Solution Gateway, the service is ready to receive notifications.
- The notifications types are described in 'OMA RCS Solution Gateway API Description' document. For this examples only the 'messageNotification' will be procesed.

| | |
|---|---|
| SERVLET HTTP METHOD EXPECTED | POST |
| REQUEST BODY<br><br>STRUCTURE<br><br>RECEIVED<br><br>(Only for messageNotification type. Refer to RCS Solution Gateway API Description document for all notifications type) | JSON<br>**messageNotification**<br>**link**<br>**rel**: MessageStatusReport<br>**href**: http://80.81.127.134:8080/rcsbox-servlet-oma/chat/0.1/OmaExample2/oneToOne/sip:385914371@solaiemes.com/-542174159/messages/1348226105779-115827579/status<br>**senderAddress**: sip:385914371@solaiemes.com<br>**chatMessage**<br>**text**: Hello world!<br>**sessionId**: -542174159<br>**messageId**: 1348226105779-115827579 |
| STATUS CODES | - 204 : Message sent OK. Not resource generate in response. Empty response<br>- 2xx : Message sent with generating a resource in the json response.<br>- 4xx : Message sent ERROR. Status code contain an error clue. |

## EXAMPLE CODE

Before starting to look at the code, it is good to have in mind these clarifications:

- The ChatNotificationSubscription object is a java object representation of the request json object needed to sent the subscription message. By the way, the ChatMessage.toString() method return the json text to sent in the request.
- To know more about subscribe information refer to RCS Solucion Gateway API Description.
- To make the http requests in the example below the org.apache 'httpclient' and com.ning 'async-http-client' are used.
- To parse json response objects the net.sf 'json-lib' is used.
- The notifications type and json structures  availables are in RCS Soluction Gateway API Description.

```
public class OmaReceiver extends HttpServlet implements ServletContextListener{


        private static final long serialVersionUID = 1L;


        //Data

        private static final String BASE_PATH = "http://192.168.0.48:8080/";

        private final String servletUrl =
        "http://192.168.0.64:8180/OMAAPIExamples/OmaReceiver";

        private final String username = "OmaExample2";


        //Constructor

        public OmaReceiver() {}


        //Get not supported

        protected void doGet(HttpServletRequest request, HttpServletResponse response)
                                    throws ServletException, IOException {

                response.sendError(405);

        }


        //Post - To receive Notifications

        protected void doPost(HttpServletRequest request, HttpServletResponse response)
                                    throws ServletException, IOException {

                //Parse json resquest to an Object

                String jsonResponse = getRequestBody(request);

                System.out.println("Request: "+jsonResponse);

                JSONObject NotificationReceived = (JSONObject)
        JSONSerializer.toJSON(jsonResponse);
```

```
            //Get 'messageNotification' data type only
            JSONObject messageNotif =
            NotificationReceived.getJSONObject("messageNotification");
            if(messageNotif != null){//See API for Notification Types
                System.out.println("Message Notification Received
                                    successfully.");
            }
            JSONObject chatMessage = messageNotif.getJSONObject("chatMessage");//Chat
                                                                message object
            String receivedMessage = chatMessage.getString("text");//Text of message
            String sender = messageNotif.getString("senderAddress");//Sender of the
                                                                message

            //Echo response
            String echoMessage = receivedMessage+ " - " +(new
                                    Date()).toString();//Echo response to send
            OMA_SendIMAutomaticSessionMessage.sendIMAutomaticMessage(username, sender,
                                                                echoMessage);
        }

        //Destroy context Listener - When the servlet is undeployed
        public void contextDestroyed(ServletContextEvent arg0) {
            OMA_Unregister.unregister(username);
        }

        //Init context Listener - When the servlet is deployed
        public void contextInitialized(ServletContextEvent arg0) {
            try {
                this.subscribe(username);
                System.out.println("Suscribed ok for user: OmaExample2."+", url:
            "+servletUrl);
            } catch (Exception e) {
                System.out.println("Error suscribing user to Chat
            Notification.");
                e.printStackTrace();
            }
        }

        //Suscribe the user into the RCS Gateway means set the Notification Url where the          user will
receive notifications.
```

```
    public void subscribe(String username) throws ClientProtocolException,
                                                                IOException {

            //Register the user

            OMA_Register.register(username);

            //Http client creation - Post method

            DefaultHttpClient httpclient = new DefaultHttpClient();

            URL url = new URL(BASE_PATH);

            HttpHost targetHost = new HttpHost(url.getHost(), url.getPort(),
                                    url.getProtocol());

            HttpPost httpPost = new HttpPost(String.format("%s%s", BASE_PATH,
                            "rcsbox-servlet-oma/chat/0.1/" + username +
                    "/subscriptions"));

            System.out.println("Request URL: POST " + httpPost.getURI());


            //Java Object that represent the json TO json String object

            ChatNotificationSubscription cns = new ChatNotificationSubscription();

            CallbackReference callback = new CallbackReference();

            callback.setNotifyURL(servletUrl);

            cns.setCallbackReference(callback);

            cns.setAdhocChatSupported(true);//Automatic IMSession

            cns.setConfirmedChatSupported(false);//Session management

            cns.setDuration(0);// value 0 mean not expires

            //print json

            System.out.println(cns.toString());


            //Set body of request

            StringEntity entity = new StringEntity(cns.toString());

            httpPost.setEntity(entity);

            HttpResponse response = httpclient.execute(targetHost, httpPost);

            //Json response of suscriber request

            String responseBody = EntityUtils.toString(response.getEntity());

            System.out.println("Subscribe response:\n" + responseBody);

    }


    //Get body text of a request object

    public String getRequestBody(HttpServletRequest request) throws IOException {

  BufferedReader reader = request.getReader();

  char[] buf = new char[4 * 1024]; // 4 KB char buffer

  StringBuilder stringBuilder = new StringBuilder();
```

```
            while (reader.read(buf, 0, buf.length) != -1) {
                stringBuilder.append(buf);
                }
                return stringBuilder.toString();
              }
}
```

## 8. Use cases

Different examples of use cases implemented using the API can be viewed at:

**Google Weather (mash-up using Google API and RCS Solution GW)**

http://www.youtube.com/watch?v=iiIcoJUNcTM

**Google Translate (mash-up using Google API and RCS Solution GW)**

http://www.youtube.com/watch?v=bt7CKoCHl_s

**Social gateway using RCS (using Twitter API, Facebook API and RCS Solution GW)**

http://www.youtube.com/watch?v=zF_roI2IOA0&feature=share&list=UUNmgKuptFwMorm6qnTSd-2Q

**Mobile Banking (demo using a mobile CRM command parser and RCS Solution GW)**

http://www.youtube.com/watch?v=9PD7lFbI9Js&feature=share&list=UUNmgKuptFwMorm6qnTSd-2Q