RCS Profile of RESTful Network APIs

Candidate Version 3.0 – 03 Feb 2015

Open Mobile Alliance

OMA-TS-REST_NetAPI_RCSProfile-V3_0-20150203-C

Contents

1. Scope

This specification provides the RCS profile of RESTful Network APIs.

The RCS profile of RESTful Network APIs specifies a subset of the existing OMA RESTful Network APIs. This specification contains tables with information on which operations are mandated in the profile that MUST be implemented in order to claim conformance with the profile and which operations are optional.

The GSMA RCS project is addressing deployment and operational considerations for 3rd party applications, and is re-using a subset of the OMA RESTful Network APIs for this. It aims to reduce the effort and time needed to create applications and to allow seamless exchange of content across mobile operators.

2. References

2.1 Normative References

[Autho4API_10]	“Authorization Framework for Network APIs”, Open Mobile Alliance™, OMA-ER-Autho4API-V1_0, URL: http://www.openmobilealliance.org/
[REQ_RCS]	“Rich Communication Suite, RCS API Detailed Requirements”, Version 2.3.1, URL: http://www.gsma.com/network2020/rcs/specs-and-product-docs/
[REST_NetAPI_ CapabilityDiscovery]	“RESTful Network API for Capability Discovery”, Version 1.0, Open Mobile Alliance™, OMA-TS-REST_NetAPI_ CapabilityDiscovery -V1_0, URL: http://www.openmobilealliance.org/
[REST_NetAPI_3PC]	“RESTful Network API for Third Party Call”, Open Mobile Alliance™, OMA-TS-REST_NetAPI_ThirdPartyCall-V1_0, URL:http://www.openmobilealliance.org/
[REST_NetAPI_ACR]	“RESTful Network API for Anonymous Customer Reference Management”, Open Mobile Alliance™, OMA-TS-REST_NetAPI_ACR-V1_0, URL: http://www.openmobilealliance.org/
[REST_NetAPI_AddressBook]	“RESTful Network API for Address Book”, Open Mobile Alliance™, OMA-TS-REST_NetAPI_AddressBook-V1_0, URL:http://www.openmobilealliance.org/
[REST_NetAPI_CallNotif]	“RESTful Network API for Call Notification”, Open Mobile Alliance™, OMA-TS-REST_NetAPI_CallNotification-V1_0, URL:http://www.openmobilealliance.org/
[REST_NetAPI_Chat]	“RESTful Network API for Chat”, Open Mobile Alliance™, OMA-TS-REST_NetAPI_Chat-V1_0, URL:http://www.openmobilealliance.org/
[REST_NetAPI_FileTransfer]	“RESTful Network API for File Transfer”, Open Mobile Alliance™, OMA-TS-REST_NetAPI_FileTransfer-V1_0, URL:http://www.openmobilealliance.org/
[REST_NetAPI_ImageShare]	“RESTful Network API for Image Share”, Open Mobile Alliance™, OMA-TS-REST_NetAPI_ImageShare-V1_0, URL:http://www.openmobilealliance.org/
[REST_NetAPI_Location]	“RESTful Network API for Terminal Location”, Version 1.0, Open Mobile Alliance™, OMA-TS-REST_NetAPI_TerminalLocation-V1_0, URL: http://www.openmobilealliance.org/
[REST_NetAPI_Messaging]	“RESTful Network API for Messaging”, Open Mobile Alliance™, OMA-TS-REST_NetAPI_Messaging-V1_0, URL:http://www.openmobilealliance.org/
[REST_NetAPI_NMS]	“RESTful Network API for Network Message Storage”, Version 1.0, Open Mobile Alliance™, OMA-TS-REST_NetAPI_NMS-V1_0, URL: http://www.openmobilealliance.org/
[REST_NetAPI_NotifChnl]	“RESTful Network API for Notification Channel”, Version 1.0, Open Mobile Alliance™, OMA-TS-REST_NetAPI_NotificationChannel-V1_0, URL: http://www.openmobilealliance.org/
[REST_NetAPI_Presence]	“RESTful Network API for Presence”, Version 1.0, Open Mobile Alliance™, OMA-TS-REST_NetAPI_Presence-V1_0, URL: http://www.openmobilealliance.org/
[REST_NetAPI_VideoShare]	“RESTful Network API for Video Share”, Version 1.0, Open Mobile Alliance™, OMA-TS-REST_NetAPI_VideoShare-V1_0, URL: http://www.openmobilealliance.org/
[REST_NetAPI_WRTCSig]	“RESTful Network API for WebRTC Signaling”, Version 1.0, Open Mobile Alliance™, OMA-TS-REST_NetAPI_WebRTCSignaling-V1_0, URL: http://www.openmobilealliance.org/
[RFC2119]	“Key words for use in RFCs to Indicate Requirement Levels”, S. Bradner, March 1997, URL:http://www.ietf.org/rfc/rfc2119.txt
[SCRRULES]	“SCR Rules and Procedures”, Open Mobile Alliance™, OMA-ORG-SCR_Rules_and_Procedures, URL:http://www.openmobilealliance.org/

2.2 Informative References

[OMADICT]	“Dictionary for OMA Specifications”, Version 2.9, Open Mobile Alliance™, OMA-ORG-Dictionary-V2_9, URL: http://www.openmobilealliance.org/

3. Terminology and Conventions

3.1 Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

All sections and appendixes, except “Scope” and “Introduction”, are normative, unless they are explicitly indicated to be informative.

3.2 Definitions

For the purpose of this TS, all definitions from OMA Dictionary [OMADICT] as well as definitions from individual RESTful Network APIs apply.

3.3 Abbreviations

ACR	Anonymous Customer Reference
API	Application Programming Interface
GSMA	GSM Association
HTTP	HyperText Transfer Protocol
NMS	Network Message Storage
OMA	Open Mobile Alliance
RCE	Rich Communication Ecosystem
RCS	Rich Communication Suite
REST	REpresentational State Transfer
SCR	Static Conformance Requirements
TS	Technical Specification
WebRTC	Web Real-Time Communication

4. Introduction

The RCS profile of RESTful Network APIs defines a subset of the resources and HTTP methods in these APIs that must be supported by any entity conforming to the profile as required by GSMA RCS [REQ_RCS]. The profile does not change the operations themselves in any way, e.g. parameters, whether optional or mandatory, behaviour, etc.

4.1 Version 3.0

Version 3.0 of the RCS Profile of RESTful Network APIs defines subsets of the following APIs:

- RESTful Network API for Image Share V 1.0
- RESTful Network API for Messaging V 1.0
- RESTful Network API for Video Share V 1.0
- RESTful Network API for File Transfer V 1.0
- RESTful Network API for Address Book V 1.0
- RESTful Network API for Chat V 1.0
- RESTful Network API for Presence V 1.0
- RESTful Network API for Notification Channel V 1.0
- RESTful Network API for Third Party Call V 1.0
- RESTful Network API for Call Notification V 1.0
- RESTful Network API for Terminal Location V 1.0
- RESTful Network API for Anonymous Customer Reference Management V1.0
- RESTful Network API for Capability Discovery V1.0
- RESTful Network API for Network Message Store V1.0
- RESTful Network API for WebRTC Signaling V1.0

as specified in the following chapter.

Following features, required by GSMA RCS [REQ_RCS], have not been implemented in this version:

- Group Chat UNI-CHT-002b, UNI-CHT-004b, UNI-CHT-007b, and UNI-CHT-008b: Allow to signal additional services such as FT and Geolocation during group chat session creation, on invitation acceptance notification, on invitation notification, and on accepting an invitation.
- Chat Media UNI-CHT-018: Add “failure” to the list of return states of sending a chat message.
- FileTransfer UNI-FLT-001: File transfer within a group chat is supported only if notified by the RCS enabler. Hence there is a need to expose querying that capability, and possibly setting it.
- FileTransfer UNI-FLT-004b and UNI-FLT-0011: The File Transfer API support indication of file transfer progress status, including indication of resumption.
- Capability Discovery UNI-CPD-003 and UNI-CPD-004: Query of service capabilities for an ad-hoc created list of contacts is not supported. Also aggregation via different notifications in the response is not supported.

5. RCS Profile of RESTful Network APIs

This section gives an overview in a form of tables of all RCS API operations and their relations to the RESTful Network APIs.

The section numbers in the column “REST Methods” refer to sections in the RESTful Network APIs specifications where more details about REST methods, operations and examples that relate to RCS operations can be found. The column “Comments” gives some clarifications to the related RCS operations.

5.1 Notification Channel

The RCS profile of the RESTful Network API for Notification Channel defines a subset of the HTTP resources/methods in [REST_NetAPI_NotifChnl] as listed below.

RCS Operations	REST Methods	Comments
Establish Notification Channel for Long Polling	6.1.5	To create Notification Channel, client application uses POST method described in 6.1.5 on the resource defined in 6.1. POST response includes ‘channel URL’ which is used to retrieve notifications using ‘Retrieve notifications from the Notification Server using Long Polling’ operation described below, and ‘callback URL’ that client application should use as notification URL when subscribing to notifications for a particular service.
Establish Notification Channel for WebSockets	6.1.5	To create Notification Channel, client application uses POST method described in 6.1.5 on the resource defined in 6.1. POST response includes ‘channel URL’ which is used to open a WebSocket connection through which the application will later receive notifications, and ‘callback URL’ that client application should use as notification URL when subscribing to notifications for a particular service. This operation is OPTIONAL.
Retrieve Notification Channel information	6.2.3	To retrieve information about Notification Channel created with ‘Establish Notification Channel for Long Polling’ operation, client application uses GET method described in 6.2.3 for the resource defined in 6.2.
Terminate Notification Channel	6.2.6	To terminate Notification Channel, client application uses DELETE method described in 6.2.6 for the resource defined in 6.2.
Retrieve notifications from Notification Server using Long Polling	6.3.5	To retrieve notifications from the Notification Server, client application uses POST method on the ‘channel URL’ received during the creation of the channel with ‘Establish Notification Channel for Long Polling’ operation.
Receive notifications from Notification Server via WebSockets	Appendix I.1	To receive notifications from the Notification Server, client application opens a WebSocket connection on the ‘channel URL’ received during the creation of the channel. This operation is REQUIRED for WebSockets-based notification channels.

Application-layer connection checking and keep-alive	Appendix I.3	To initiate connection checking and keep-alive, client application sends a "connCheck" message to the server, using the WebSockets connection of the notification channel. The server responds with a "connAck" message. This operation is REQUIRED for WebSockets-based notification channels.
Refresh Notification Channel	6.4.4	To refresh a Notification Channel (i.e. to update its lifetime), the client application uses PUT method described in 6.4.4 on the resource defined in 6.4 This operation is REQUIRED for WebSockets-based notification channels, otherwise OPTIONAL.

5.2 Chat

The RCS Profile of the RESTful Network API for Chat defines a subset of the HTTP resources/methods in [REST_NetAPI_Chat] as listed below.

RCS Operations	REST Methods	Comments
Subscribe for notifications on updates in a chat session	6.1.5	To create a subscription for notifications on updates in a chat session (for example: new events, session invitation, participant status, and message delivery status), client application uses POST method described in 6.1.5 on the resource defined in 6.1. Note that for client applications that cannot maintain notification URL, the notification URL SHALL be previously obtained by executing the operation 'Establish Notification Channel for Long Polling' (section 5.1)
Cancel subscription for notifications on updates in a chat session	6.2.6	To cancel a subscription for notifications on updates in a chat session, client application uses DELETE method described in 6.2.6 on the resource defined in 6.2.

Recieve notifications on updates in a chat session	6.16.5, 6.17.5, 6.18.5, 6.19.5, 6.20.5, 6.21.5	Receive notifications on updates in a chat session at the notification URL provided when executing 'Subscribe for notifications on updates in a chat session' operation. For incoming messages, as described in section 6.16.5. For notifications about message delivery status, as described in section 6.17.5 For notifications about a 1-1 chat session invitation, as described in section 6.18.5. For notifications about a group chat session invitation, as described in section 6.19.5. For notifications about chat session events, as described in section 6.20.5. For notifications about changes in group chat participant status, as described in section 6.21.5. Note that for client applications that cannot maintain a notification URL, the updates SHALL be obtained by executing the 'Retrieve notifications from Notification Server using Long Polling' operation (section 5.1).
Create (initiate) a 1-1 chat session	6.3.5	To create (initiate) a 1-1 chat session, client application uses POST method described in 6.3.5 on the resource defined in 6.3.
Accept a 1-1 chat invitation	6.5.4	To accept an inivitation for a 1-1 chat session, client application uses PUT method described in 6.5.4 on the resource defined in 6.5.
Decline a 1-1 chat session invitation	6.4.6	To decline a 1-1 chat session invitation, the invited client application uses DELETE method described in 6.4.6 on the resource defined in 6.4.
Cancel a 1-1 chat session invitation	6.4.6	To cancel a group chat session invitation, the inviting client application uses DELETE method described in 6.4.6 on the resource defined in 6.4. To have an effect, the operation must be performed before the invitation has been accepted by some of the invited participants.
Extend a 1-1 chat session to a group chat session	6.6.5	To extend a 1-1 chat session to a group chat session, client application uses POST method described in 6.6.5 on the resource defined in 6.6.
Create (send) 1-1 chat message, including 'isComposing'	6.7.5	To create (send) a new 1-1 chat message, client application uses POST method described in 6.7.5 on the resource defined in 6.7. The entity body of the POST request is either a plain chat message, or a specific 'isComposing' message.
Report message delivery status in a 1-1 chat	6.8.4	To report a group chat message delivery status, client application uses PUT method described in 6.8.4 on the resource defined in 6.8.
Create (initiate) group chat session	6.9.5	To create (initiate) a group chat delivery, client application uses POST method described in 6.9.5 on the resource defined in 6.9.
Accept group chat invitation	6.13.4	To accept an inivitation for a group chat session, client application uses PUT method described in 6.13.4 on the resource defined in 6.13.
Decline a group chat session invitation	6.12.6	To decline a group chat session invitation, the invited client application uses DELETE method described in 6.12.6 on the resource defined in 6.12.

Cancel a group chat session invitation	6.10.6	To cancel a group chat session invitation, the inviting client application uses DELETE method described in 6.10.6 on the resource defined in 6.10. To have an effect, the operation must be performed before the invitation has been accepted by some of the invited participants.
Create (send) group chat message	6.14.5	To create (send) a new group chat message, client application uses POST method described in 6.14.5 on the resource defined in 6.14.
Add one or more participants to a group chat session	6.11.5	To add one or more participants to the esatblished group chat session, client application uses POST method described in 6.11.5 on the resource defined in 6.11. Note that it depends on a server policy who can add new participants to a group chat session (any Participant or just the Originator)
Remove particiant(s) from a group chat session	6.12.6	To remove one or more participants from the esatblished group chat session, client application uses DELETE method described in 6.12.6 on the resource defined in 6.12. Note that it depends on a server policy who can remove participant(s) from a group chat session (any Participant or just the Originator)
Leave group chat session	6.12.6	To leave a group chat session, client application uses DELETE method described in 6.12.6 on the resource defined in 6.12.Note that it depends on a server policy what will happen with the group chat session if the Originator leaves the session (e.g. to terminate session or continue the session with the rest of participants).
Re-join a group chat session	6.11.5	To re-join the esatblished group chat session, client application uses POST method described in 6.11.5 on the resource defined in 6.11
Terminate a group chat session	6.10.6	To terminate a group chat session, client application uses DELETE method described in 6.10.6 on the resource defined in 6.10
Create (initiate) long lived group chat	6.9.5	To create (initiate) a long lived group chat, client application uses POST method described in 6.9.5 on the resource defined in 6.9.
Accept long lived group chat invitation	6.13.4	To accept an inivitation for a long lived group chat, client application uses PUT method described in 6.13.4 on the resource defined in 6.13.
Decline a long lived group chat invitation	6.12.6	To decline a long lived group chat session invitation, the invited client application uses DELETE method described in 6.12.6 on the resource defined in 6.12.
Cancel a long lived group chat invitation	6.10.6	To cancel a long lived group chat session invitation, the inviting client application uses DELETE method described in 6.10.6 on the resource defined in 6.10. To have an effect, the operation must be performed before the invitation has been accepted by some of the invited participants.
Create (send) chat message in a long lived group chat	6.14.5	To create (send) a new long lived group chat message, client application uses POST method described in 6.14.5 on the resource defined in 6.14.

Add one or more participants to a long lived group chat	6.11.5	To add one or more participants to the established long lived group chat, client application uses POST method described in 6.11.5 on the resource defined in 6.11. Note that it depends on a server policy who can add new participants to a long lived group chat (any Participant or just the Originator)
Remove particiant(s) from a long lived group chat	6.12.6	To remove one or more participants from the established long lived group chat, client application uses DELETE method described in 6.12.6 on the resource defined in 6.12. Note that it depends on a server policy who can remove participant(s) from a long lived group chat (any Participant or just the Originator)
Leave long lived group chat	6.12.6	To leave a long lived group chat, client application uses DELETE method described in 6.12.6 on the resource defined in 6.12.Note that it depends on a server policy what will happen with the group chat session if the Originator leaves the session (e.g. to terminate session or continue the session with the rest of participants).
List all long lived group chat	6.9	Get the list of all long lived group chat the user is participating to (ie has been invited or has already accepting the invitation).

Note: Multimedia chat messages (UNI-CHT-026, -027) is part of the requirements but not supported yet by the underlying enabler.

NOTE: Same methods are used to ensure Group Chat and Long Lived Group Chat requirements. As a consequence it is assumed that an instance of the RCS Network API gateway will not have to handle Group Chat and Long Lived Group Chat for the same user. This gateway implements the chosen Group chat mode (either Group Chat or Long Lived Group Chat) towards the RCS network and complies with the associated RCS User Network Intefarce.

When configured to implement Long Lived Group chat, the server (gateway) will expose to the client any Long Lived Group chat session the user has been participating to as an active session. The gateway stores all the necessary context information to comply with the Long Lived Group chat call RCS call flows. E.g. In case new messages sent for a specific Long Lived Group chat and the MSRP media flow has been torn down due to media inactivité, the gateway will implement the rejoin/restart procedures of RCS.

5.3 File Transfer

The RCS Profile of the RESTful Network API for FileTransfer defines a subset of the HTTP resources/methods in [REST_NetAPI_FileTransfer] as listed below.

RCS Operations	**REST Methods**	**Comments**
Create a new 1-1 file transfer session	6.3.5	To create a 1-1 file transfer session, client application uses POST method described in 6.3.5 on the resource defined in 6.3. The operation can either include the actual file content or just external file repository URL The operation can support multi-files in one session,; for RCS only one file is supported.

Cancel file transfer invitation	6.4.6	To cancel a file transfer session invitation, client application uses DELETE method described in 6.4.6 on the resource defined in 6.4. When the file transfer session status is “Invited”, the application of originator can use this operation to cancel file transfer invitation.
End the file transfer session	6.4.6	To end a file transfer session, client application uses DELETE method described in 6.4.6 on the resource defined in 6.4. When the file transfer session status is ”Connected”, both the application of originator and the application of receiver can use this operation to terminate the file transfer session.
Create subscription for file transfer notifications	6.1.5	To create a subscription for event notifications in a file transfer session, client application uses POST method described in 6.1.5 on the resource defined in 6.1. Note that for client applications that cannot maintain notification URL, the notification URL SHALL be previously obtained by executing the operation ‘Establish Notification Channel for Long Polling’ (section 5.1)
Cancel subscription for file transfer notifications	6.2.6	To cancel a subscription for file transfer events notifications, the client application uses the DELETE method described in 6.2.6 on the resource defined in 6.2.
Notifications about File Transfer event (declined, cancelled, ended) and MSRP transfer session state “success”, “abort” and “error”)	6.8.5	Receive notifications about file transfer session events at the notification URL provided when executing ‘Create subscription for file transfer notifications’ operation- Events covered by those notifications include SessionCancelled, SessionEnded, Declined, Successful, Failed, Aborted. Note that for client applications that cannot maintain a notification URL, the updates SHALL be obtained by executing the ‘Retrieve notifications from Notification Server using Long Polling’ operation (section 5.1).
Notification about receiver acceptance	6.10.5	Receive notifications about receiver acceptance for the file transfer seesion invitation at the notification URL provided when executing ‘Create subscription for file transfer notifications’ operation.. If the receiver accepted the file transfer, the notification includes the status,which is “Connected”, and for RCS there should be only one file accepted.. Note that for client applications that cannot maintain a notification URL, the updates SHALL be obtained by executing the ‘Retrieve notifications from Notification Server using Long Polling’ operation (section 5.1).
Notification about file transfer invitation	6.7.5	Receive notifications about file transfer session invitations at the notification URL provided when executing ‘Create subscription for file transfer notifications’ operation. In SessionInvitationNotificaiton, for RCS there should be only one file supported. Note that for client applications that cannot maintain a notification URL, the updates SHALL be obtained by executing the ‘Retrieve notifications from Notification Server using Long Polling’ operation (section 5.1).

Accept a file transfer invitation	6.5.4	To accept an invitation for file transfer session, client application uses PUT method described in 6.5.4 on the resource defined in 6.5. In ReceiverSessionStatus, for RCS there should be only one file supported.
Decline a file transfer invitation	6.4.6	To decline an invitation for file transfer session, client application uses DELETE method described in 6.4.6 on the resource defined in 6.4. When the file transfer session status is “Invited”, the application of receiver can use this operation to decline file transfer invitation.
Notification about file content link	6.9.5	Receive notifications about the link to file content at the notification URL provided when executing ‘Create subscription for file transfer notifications’ operation. In FileNotification, for RCS there should be only one file supported.. Note that for client applications that cannot maintain a notification URL, the updates SHALL be obtained by executing the ‘Retrieve notifications from Notification Server using Long Polling’ operation (section 5.1).
Notification about subscription cancellation	6.11.5	Receive notifications about subscription cancellations at the notification URL provided when executing ‘Create subscription for file transfer notifications’ operation. Note that for client applications that cannot maintain a notification URL, the updates SHALL be obtained by executing the ‘Retrieve notifications from Notification Server using Long Polling’ operation (section 5.1).

5.4 Third Party Call

The RCS Profile of the RESTful Network API for Third Party Call defines a subset of the HTTP resources/methods in [REST_NetAPI_3PC] as listed below.

RCS Operations	REST Methods	Comments
Initiate call session	6.1.5	To create call session, client application uses POST method described in 6.1.5 on the resource defined in 6.1
Add a participant to a call session	6.4.5	To add a participant to a call session, client application uses POST method described in 6.4.5 on the resource defined in 6.4
Remove a participant from a call session	6.5.6	To remove a participant to a call session, client application uses DELETE method described in 6..5.6 on the resource defined in 6.5
Retrieve call session information	6.2.3	To retrieve call session information, client application uses GET method described in 6.2.3 on the resource defined in 6.2
Retrieve information about all particiopants in a call session	6.4.3	To retrieve information about all participant in a call session, client application uses GET method described in 6.4.3 on the resource defined in 6.4

Retrieve information about individual particiopant in a call session	6.5.3	To retrieve information about individual participant in a call session, client application uses GET method described in 6.5.3 on the resource defined in 6.5
Terminate or cancel call session	6.2.6	To terminate a call session or cancel a call session attempt, client application uses DELETE method described in 6.2.6 on the resource defined in 6.2

5.5 Call Notification

The RCS Profile of the RESTful Network API for Call Notification defines a subset of the HTTP resources/methods in [REST_NetAPI_CallNotif] as listed below.

RCS Operations	REST Methods	Comments
Subscribe to notifications about call events	6.2.5	To subscribe to notifications about call events, client application uses POST method described in 6.2.5 on the resource defined in 6.2 Section 5.2.3.1 gives details about the call events the client can subscribe for. Note that for client applications that cannot maintain notification URL, the notification URL SHALL be previously obtained by executing the operation ‘Establish Notification Channel for Long Polling’ (section 5.1)
Retrieve an individual subscribtion to notifications about call events	6.3.3	To retrieve an individual subscribtion to notifications about call events, client application uses GET method described in 6.3.3 on the resource defined in 6.3
Cancel an individual subscribtion to notifications about call events	6.3.6	To cancel (delete) an individual subscribtion to notifications about call events, client application uses DELETE method described in 6.3.6 on the resource defined in 6.3
Subscribe to notifications about play-and-collect-media interraction	6.6.5	To subscribe to notifications about play-and-collect-media interraction, client application uses POST method described in 6.6.5 on the resource defined in 6.6.
Retrieve an individual subscribtion to notifications about play-and-collect-media interraction	6.7.3	To retrieve an individual subscribtion to notifications about play-and-collect-media interraction, client application uses GET method described in 6.7.3 on the resource defined in 6.7.
Cancel an individual subscribtion to notifications about play-and-collect-media interraction	6.7.6	To cancel (delete) an individual subscribtion to notifications play-and-collect-media interraction, client application uses DELETE method described in 6.7.6 on the resource defined in 6.7.
Retrieve notifications about call events	6.10.5	Receive notifications about call events at the notification URL provided when executing ‘Subscribe to notifications about call events’ operation. Note that for client applications that cannot maintain a notification URL, the updates SHALL be obtained by executing the ‘Retrieve notifications from Notification Server using Long Poling’ operation (section 5.1).

Retrieve notifications about play-and-collect media interaction	6.11.5	Receive notifications about play-and-collect media interaction at the notification URL provided when executing ‘Subscribe to notifications about play-and-collect media interaction’ operation. Note that for client applications that cannot maintain a notification URL, the updates SHALL be obtained by executing the ‘Retrieve notifications from Notification Server using Long Poling’ operation (section 5.1).

5.6 Video Share

The RCS Profile of the RESTful Network API for Video Share defines a subset of the HTTP resources/methods in [REST_NetAPI_VideoShare] as listed below.

RCS Operations	REST Methods	Comments
Create subscription to video share notifications	6.1.5	Create a new subscription to video share notifications. Note that for client applications that cannot maintain notification URL, the notification URL SHALL be previously obtained by executing the operation ‘Establish Notification Channel for Long Polling’ (section 5.1).
Cancel subscription to video share notifications	6.2.6	Cancel a subscription and stop corresponding notifications.
Create a new 1-1 video share session	6.3.5	Create a new 1-1 video share session with CS call related or without CS call related. The operation supports recorded video and live video.
Cancel a 1-1 video share session invitation	6.4.6	Cancel a 1-1 video share session invitation before the invitation has been accepted.
Decline a 1-1 video share session invitation	6.4.6	Decline a 1-1 video share session invitation before the invitation has been accepted.
End a 1-1 video share session	6.4.6	Terminate a 1-1 video share session after the session has been in connected status.
Accept a 1-1 video share session invitation	6.5.4	Accept a 1-1 video share session invitation with supported media format and receive the media URL for accessing the video content.
Receive notifications for video share invitations	6.6.5	Receive notifications for video share session invitations at the notification URL provided when executing ‘Create subscription to video share notifications’ operation. Note that for client applications that cannot maintain a notification URL, the updates SHALL be obtained by executing the ‘Retrieve notifications from Notification Server using Long Polling’ operation (section 5.1).

Receive notifications for video share session acceptance	6.7.5	Receive notifications for video share session invitation acceptance at the notification URL provided when executing 'Create subscription to video share notifications' operation. Note that for client applications that cannot maintain a notification URL, the updates SHALL be obtained by executing the 'Retrieve notifications from Notification Server using Long Polling' operation (section 5.1).
Receive notifications for video share events	6.8.5	Receive notifications for video share session events at the notification URL provided when executing 'Create subscription to video share notifications' operation. Note that for client applications that cannot maintain a notification URL, the updates SHALL be obtained by executing the 'Retrieve notifications from Notification Server using Long Polling' operation (section 5.1).

5.7 Image Share

The RCS Profile of the RESTful Network API for Image Share defines a subset of the HTTP resources/methods in [REST_NetAPI_ImageShare] as listed below.

RCS Operations	REST Methods	Comments
Create subscription to image share notifications	6.1.5	Create a new subscription to image share notifications. Note that for client applications that cannot maintain notification URL, the notification URL SHALL be previously obtained by executing the operation 'Establish Notification Channel for Long Polling' (section 5.1).
Cancel subscription to image share notifications	6.2.6	Cancel asubscription and stop corresponding notifications.
Create a new image share session	6.3.5	Create a new image share session with CS call related or without CS call related. The operation can either include the actual image file content or include just a file repository URL via which the image file content can be retrieved,
Cancel an image share invitation	6.4.6	Cancel an image share session invitiation before the invitation has been accepted.
Decline an image share invitation	6.4.6	Decline an image share session invitation before the invitation has been accepted.
End an image share session	6.4.6	Terminate an image share session after the session has been in conncted status.
Accept an image share invitation	6.5.4	Accept an image share session invitation.

Receive notifications for image share invitations	6.6.5	Receive notifications for image share session invitations at the notification URL provided when executing 'Create subscription to image share notifications' operation. Note that for client applications that cannot maintain a notification URL, the updates SHALL be obtained by executing the 'Retrieve notifications from Notification Server using Long Polling' operation (section 5.1)..
Receive notification s for image share session acceptance	6.7.5	Receive notifications for image share session invitation acceptance at the notification URL provided when executing 'Create subscription to image share notifications' operation. Note that for client applications that cannot maintain a notification URL, the updates SHALL be obtained by executing the 'Retrieve notifications from Notification Server using Long Polling' operation (section 5.1).
Receive notifications for links to the image file	6.8.5	Receive notifications for links to the image file at the notification URL provided when executing 'Create subscription to image share notifications' operation. Client applications can use the link to download the image file. Note that for client applications that cannot maintain a notification URL, the updates SHALL be obtained by executing the 'Retrieve notifications from Notification Server using Long Polling' operation (section 5.1).
Receive notifications for image share events and image file delivery status.	6.9.5	Receive notifications for image share session events and image file delivery status at the notification URL provided when executing 'Create subscription to image share notifications' operation. Note that for client applications that cannot maintain a notification URL, the updates SHALL be obtained by executing the 'Retrieve notifications from Notification Server using Long Polling' operation (section 5.1).

5.8 Messaging

The RCS Profile of the RESTful Network API for Messaging defines a subset of the HTTP resources/methods in [REST_NetAPI_Messaging] as listed below.

RCS Operations	REST Methods	Comments
Send message	6.9.5	Create an outbound message request. To send an SMS or MMS, to one or multiple destination address(es). To request delivery confirmation via a notification, the client application must provide a notification URL when executing this operation. Note that for client applications that cannot maintain notification URL, the notification URL SHALL be previously obtained by executing the operation 'Establish Notification Channel for Long Polling' (section 5.1).

Receive delivery confirmation	6.14.5	Receive delivery confirmation (“delivered”, and/or “displayed”) at the notification URL provided when executing ‘Send message’ operation. Note that for client applications that cannot maintain a notification URL, the updates SHALL be obtained by executing the ‘Retrieve notifications from Notification Server using Long Polling’ operation (section 5.1).
Subscribe to receive messages	6.6.5	Create a subscription to receive messages. Note that for client applications that cannot maintain notification URL, the notification URL SHALL be previously obtained by executing the operation ‘Establish Notification Channel for Long Polling’ (section 5.1).
Receive message(s)	6.8.5	Receive message(s) at the notification URL provided when executing ‘Subscribe to receive messages’ operation MUST be executed once before this operation. Note that for client applications that cannot maintain a notification URL, the updates SHALL be obtained by executing the ‘Retrieve notifications from Notification Server using Long Polling’ operation (section 5.1).
Send “displayed” notification for received message	6.15.4	To notify the server that the received message has been displayed on the receiver’s terminal, the receiver’s application uses PUT method described in 6.15.4 on the resource provided by the server in the “link” element of the received message.

5.9 Network Address Book

The RCS profile of the RESTful Network API for Address Book defines a subset of the HTTP resources/methods in [REST_NetAPI_AddressBook] as listed below.

RCS Operations	REST Methods	Comments
Retrieve list of contacts in the NAB	6.1.3	Retrieve list of contacts, optionally with all attributes or selected attributes. Note: NAB is instantiated when the first contact is added.
Retrieve all information for a specified contact in the VCard format	6.2.3	Retrieve information for a contact, optionally with selected attributes. Note: VCard 2.1 and VCard3.0 are retrieved as opaque objects.
Add a new contact to NAB	6.2.4	Create a new contact with or without selected attributes (e.g. VCard).
Update a contact’s information	6.2.4, 6.2.6	Update a contact’s attributes (6.2.4) or delete a contact (6.2.6). Note: the information supplied MUST represent the complete desired data for the contact, since it will completely replace the existing data for the contact in the resource. If the client does not cache the current data for the contact, then it is recommended to first retrieve the data for the contact and modify the data as appropriate, before executing this operation.

Subscribe to updates to contacts in NAB	6.15.5	Create a subscription to updates in NAB. MAY include adding and/or removing a contact and/or changing an existing contact's attributes. Note that for client applications that cannot maintain notification URL, the notification URL SHALL be previously obtained by executing the operation 'Establish Notification Channel for Long Polling' (section 5.1).
Receive notifications regarding updates to contacts in NAB	6.17.5	Receive notifications with updates at the notification URL provided when executing 'Subscribe to updates to contacts in NAB' operation. Note that for client applications that cannot maintain a notification URL, the updates SHALL be obtained by executing the 'Retrieve notifications from Notification Server using Long Polling)' operation (section 5.1).
Remove subscription	6.16.6	Remove subscription as a temporary resource.

5.10 Anonymous Customer Reference Management

The RCS Profile of the RESTful Network API for Anonymous Customer Reference Management defines a subset of the HTTP resources/methods in [REST_NetAPI_ACR] as listed below.

RCS Operations	REST Methods	Comments
Retrieve the already issued app-specific ACR	6.1.3	To retrieve the already issued application-specific ACR for the given end user, client application uses GET method described in 6.1.3 on the resource defined in 6.1
Requesting ACR associated with a particular mobile telephone number (MSISDN).	6.1.5	To issue (create) an application-specific ACR for the given user identified by the userId, client application uses POST method described in 6.1.5 on the resource defined in 6.1
Retrieves the ACR data	6.2.3	To retrieve the ACR data (value, status, expiry), client application uses GET method described in 6.2.3 on the resource defined in 6.2
Delete (remove) an ACR	6.2.6	To remove an ACR with data, client application uses DELETE method described in 6.2.6 for the resource defined in 6.2.
Query the ACR status of an user	6.3.3	To retrieve the status of an ACR, client application uses GET method described in 6.3.3 for the resource defined in 6.3.
Refresh the user's ACR	6.3.4	To refresh an "expired" ACR (i.e. set the ACR status to "valid"), client application uses PUT method described in 6.3.4 on the resource defined in 6.3.

5.11 Capability Discovery

The RCS Profile of the RESTful Network API for Capability Discovery defines a subset of the HTTP resources/methods in [REST_NetAPI_ CapabilityDiscovery] as listed below.

RCS Operations	REST Methods	Comments
Register a new service capability	6.1.5, 6.2.4, 6.3.4	To be able to register a new own service capability, a client application MUST ensure that Capability Source for that particular device/application is created. To create a new Capability Source, the client application uses POST method described in 6.1.5 on the resource defined in 6.1. With the same operation for creating the Capability Source, the application can also register one or more service capabilities for that particular Capability Source. In case the Capability Source has already been created, to register new service capability(ies) the client application SHALL use PUT method described in 6.2.4 on the resource defined in 6.2. Alternatively, for registering a single service capability, client application can use PUT method described in 6.3.4 on the Light-weight Resource defined in 6.3. Note that this is an optional feature. Note that in case a user is registering service capabilities from multiple devices/applications (subject to server policy), in order to manage service capabilities independently for each device/application, the user SHALL create a separate Capability Source for each device/application.
Deregister a previously registered service capability	6.2.4, 6.3.6, 6.2.6	To deregister (unregister) previously registered own service capability (ies) for a particular Capability Source, client application uses PUT method described in 6.2.4 on the resource defined in 6.2. Alternatively, for deregistering a single service capability, client application can use DELETE method described in 6.3.6 on the Light-weight Resource defined in 6.3. Note that this is an optional feature. Note: To deregister all service capabilities for a particular Capability Source and at the same time remove the Capability Source completely, client application uses DELETE method described in 6.2.6 on the resource defined in 6.2. When performing this operation the capabilities should be disabled first.
Enable or disable a previously registered service capability	6.2.4, 6.3.4	To enable or disable previously registered own service capability (ies) for a particular Capability Source, client application uses PUT method described in 6.2.4 on the resource defined in 6.2. Alternatively, for enabling or disabling a single service capability, client application can use PUT method described in 6.3.4 on the Light-weight Resource defined in 6.3. Note that this is an optional feature.
Query service capabilities for a contact	6.4.3	To query for a service capabilities for a contact, client application uses GET method described in 6.4.3 for the resource defined in 6.4.
Query if a certain contact is RCS capable or not.	6.4.3	To query if a certain contact is RCS capable or not, client application uses GET method described in 6.4.3 for the resource defined in 6.4. In this case, guery string parameter 'userTypeFilter' SHALL be used and its value SHALL be set to "RCS".

Retrieve registered own service capabilities	6.1.3, 6.2.3, 6.3.3	To retrieve all own service capabilities (registered for all Capability Sources), client application used GET method described in 6.1.3 on the resource defined in 6.1. To retrieve own service capabilities registered for a particular Capability Source, client application uses GET method described in 6.2.3 on the resource defined in 6.2. Alternatively, to retrieve a single own service capability registered for a particular Capability Source, client application uses GET method described in 6.3.3 on the Light-weight Resource defined in 6.3. Note that this is an optional feature.

5.12 Terminal Location

The RCS Profile of the RESTful Network API for Terminal Location defines a subset of the HTTP resources/methods in [REST_NetAPI_Location] as listed below.

RCS Operations	REST Methods	Comments
Retrieve the terminal location	6.1.3	To retrieve location for single terminal, client application uses GET method described in 6.1.3 on the resource defined in 6.1.

5.13 Presence

The RCS Profile of the RESTful Network API for Presence defines a subset of the HTTP resources/methods in [REST_NetAPI_Presence] as listed below. Note that for some RCS operations related to Presence, HTTP resources/methods are defined in [REST_NetAPI_AddressBook] which is indicated in "REST Methods" column.

RCS Operations	REST Methods	Comments
Manage "free text" presence attribute	6.4.3, 6.4.4, 6.4.6	This operation is used to manage own "free text" social presence attribute. Client application can use GET (retrieve, 6.4.3), PUT (set or update, 6.4.4) and DELETE (remove, 6.4.6) methods defined for the resource in section 6.4. Note that the resource in 6.4 provides access to complete presence attributes data structures, and the "free text" attribute is represented by element 'noteList' from data type 'PersonAttributes' in section 5.2.2.4.

Manage “portrait icon” presence attribute	6.29.3, 6.29.4, 6.29.6	This operation is used to manage own “portrait icon” social presence attribute. The resource that client application shall use is defined in section 6.29. To retrieve “portrait icon”, client application uses GET method in section 6.29.3. To set or update “portrait icon”, client application uses PUT method in section 6.29.4. This operation will simultaneously upload “portrait icon” and set the link to the “portrait icon” as presence information. To remove “portrait icon” social presence attribute, client application uses DELETE method in section 6.29.6.
Manage “favourite link” presence attribute	6.4.3, 6.4.4, 6.4.6	This operation is used to manage own “favourite link” social presence attribute. Client application can use GET (retrieve 6.4.3), PUT (set or update, 6.4.4) and DELETE (remove, 6.4.6) methods defined for the resource in section 6.4. Note that the resource in 6.4 provides access to complete presence attributes data structures, and the “favourite link” attribute is represented by element ‘linkList’ from data type ‘PersonAttributes’ in section 5.2.2.4.
Manage “location” presence attribute	6.4.3, 6.4.4, 6.4.6	This operation is used to manage own “location” social presence attribute. Client application can use GET (retrieve, 6.4.3), PUT (set or update, 6.4.4) and DELETE (remove, 6.4.6) methods defined for the resource in section 6.4. Note that the resource in 6.4 provides access to complete presence attributes data structures, and depending on the format of “location” data, the following elements from data type ‘PersonAttributes’ in section 5.2.2.4 can be used to represent the “location” attribute: - ‘placeType’, when location expressed in a form of plain text, - ‘location’ when location expressed with coordinate values, - ‘timeOffset’ when location expressed as time offset from UTC.
Manage “availability status” presence attribute	6.4.3, 6.4.4, 6.4.6	This operation is used to manage own “availability status” social presence attribute. Client application can use GET (retrieve, 6.4.3), PUT (set or update, 6.4.4) and DELETE (remove, 6.4.6) methods defined for the resource in section 6.4. Note that the resource in 6.4 provides access to complete presence attributes data structures, and the “availability status” attribute is represented by element ‘overridingWillingness’ from data type ‘PersonAttributes’ in section 5.2.2.4.
Manage multiple presence attributes as a set	6.4.3, 6.4.4, 6.4.6	This operation is used to manage multiple social presence attributes. Client application can use GET (retrieve, 6.4.3), PUT (set or update, 6.4.4) and DELETE (remove, 6.4.6) methods defined for the resource in section 6.4.

Invite a member to share presence information	6.10.4 from [REST_NetAPI_AddressBook]	Invitation of member to share presence information is done by adding a user (member) to a member list (e.g. 'rcs' list). To add a user (member) on the list, client applications use PUT method as described in [REST_NetAPI_AddressBook], section 6.10.4. However this action itself will not trigger presence sharing invitation towards the invited user unless: - the inviting user has subscribed for notifications on presence information for that particular member list, as described in 'Subscribe for notifications on presence information updates for a contact list' operation, or - the inviting user attempts to retrieve presence information for that particular member list, as described in 'Retrieve presence information for a contact list' Note that in order to see the invitation, the invited user must either subscribe for notifications by executing 'Subscribe for notifications on presence sharing invitation' operation or to retrieve pending invitations by executing 'Retrieve pending invitations' operation.
Cancel presence invitation	6.10.6 from [REST_NetAPI_AddressBook]	To cancel presence sharing invitation before the invitation is accepted, client applications use DELETE method as described in [REST_NetAPI_AddressBook], section 6.10.6
Retrieve presence information for a single contact	6.13.3	To retrieve social presence information for a given contact, client application uses GET method described in section 6.13.3 on the resource defined in 6.13.
Retrieve presence information for a contact list	6.15.3	To retrieve social presence information for all contacts (members) in a contact list (Presence List stored on the server), client application uses GET method described in section 6.15.3 on the resource defined in 6.15
Retrieve presence information for an ad-hoc created list of contacts	6.30.5	To retrieve social presence information for all contacts specified in an ad-hoc created list of contacts, client application uses POST method described in section 6.30.5 on the resource defined in 6.30.
Subscribe for notifications on presence sharing invitation	6.18.5	To subscribe for notifications for presence sharing invitation, client application uses POST method described in section 6.18.5 on the resource defined in 6.18. Note that for client applications that cannot maintain notification URL, the notification URL SHALL be previously obtained by executing the operation 'Establish Notification Channel for Long Polling' operation (section 5.1).
Retrieve notifications for presence sharing invitation	6.20.5	Receive notifications for presence sharing invitations at the notification URL provided when executing 'Subscribe for presence sharing invitation' operation. Note that for client applications that cannot maintain a notification URL, the updates SHALL be obtained by executing the 'Retrieve notifications from Notification Server using Long Poling' operation (section 5.1).

Manage presence sharing invitation	6.10.4, 6.27.5 from [REST_NetAPI_AddressBook]	To accept presence sharing invitation from a user, client application uses PUT method described in [REST_NetAPI_AddressBook], section 6.10.4, to add the user to his/her list (e.g.'rcs' list). To block presence sharing invitation from a user, client application uses PUT method described in [REST_NetAPI_AddressBook], section 6.10.4, to add the user to his/her 'blockedcontacts' list. To revoke presence sharing invitation from a user, client application uses POST method described in [REST_NetAPI_AddressBook], section 6.27.5, and transfer the user from one list to another list (e.g. from 'rcs' list to 'rcs_revokedcontacts' list). Note these operations are used to manage presence sharing relationships only, not as response to invitations for presence sharing. To ignore presence sharing invitation, no specific action is performed by client application. Note that in order to see the invitation, client application (invited user) must either subscribe for notifications by executing 'Subscribe for notifications on presence sharing invitation' operation, or to retrieve pending invitations by executing 'Retrieve pending invitations' operation.
Retrieve presence information for own Presentity	6.13.3	To retrieve presence information for the own Presentity, client application uses GET method described in 6.13.3 for the resource defined in 6.13.
Subscribe for notifications on presence information updates for own Presentity	6.22.5	To subscribe for notifications to updates on presence information for own Presentity, client application uses POST method described in section 6.22.5 on the resource defined in 6.22. Note that for client applications that cannot maintain notification URL, the notification URL SHALL be previously obtained by executing the operation 'Establish Notification Channel for Long Polling' (section 5.1).
Subscribe for notifications on presence information updates for a contact list	6.26.5	To subscribe for notifications to updates in presence information for a contact list (Presence List stored on the server), client application uses POST method described in section 6.26.5 on the resource defined in 6.26. Note that for client applications that cannot maintain notification URL, the notification URL SHALL be previously obtained by executing the operation 'Establish Notification Channel for Long Polling' (section 5.1).
Retrieve notifications on presence information updates for own Presentity	6.24.5	Receive notifications for presence sharing invitations at the notification URL provided when executing 'Subscribe for updates on presence information for own Presentity' operation. Note that for client applications that cannot maintain a notification URL, the updates SHALL be obtained by executing the 'Retrieve notifications from Notification Server using Long Poling' operation (section 5.1).
Retrieve notifications on presence information updates for a contact list	6.28.5	Receive notifications for presence sharing invitations at the notification URL provided when executing 'Subscribe for updates on presence information for a contact list' operation. Note that for client applications that cannot maintain a notification URL, the updates SHALL be obtained by executing the 'Retrieve notifications from Notification Server using Long Poling' operation (section 5.1).

Retrieve pending invitations	6.8.3	To retrieve pending invitations, client application uses GET method described in section 6.8.3 on the resource defined in 6.8.
Retrieve service capabilities	6.13.3	To retrieve service capabilities (either for own Presentity or for a contact), client application uses GET method described in section 6.13.3 on the resource defined in 6.13, with query parameter set to “?presenceFilter=service/*/*” (without quotation marks)
Retrieve service capabilities for a contact list (ad-hoc created list of contacts)	6.30.5	To retrieve service capabilities for a contact list (Presence List stored on the server), client application uses GET method described in section 6.30.5 on the resource defined in 6.30, with query parameter set to “?presenceFilter=service/*/*” (without quotation marks)
Retrieve service capabilities for a contact list (Presence List stored on the server)	6.15.3	To retrieve service capabilities for a contact list (Presence List stored on the server), client application uses GET method described in section 6.15.3 on the resource defined in 6.15, with query parameter set to “?presenceFilter=service/*/*” (without quotation marks)
Publish own service capabilities	6.1.5	To publish own service capabilities, client application uses POST method described in 6.1.5 for the resource defined in 6.1. Data structure 5.2.2.5 describes service capabilities for a particular service.

5.14 NMS

The RCS Profile of the RESTful Network API for Network Message Store defines a subset of the HTTP resources/methods in [REST_NetAPI_NMS] as listed below.

Implementations of the RCS profile MUST support a single Root folder where its name is an empty string. The Root folder MUST contain the following two system-defined subfolders: Default and RCSMessageStore.

RCS Operations	REST Methods	Comments
Create object(s) in Network Storage	6.1.5 6.9.5	To create (i.e. store, upload) a new object in the Network Storage, client application uses POST method described in 6.1.5 on the resource defined in 6.1. In order to create (i.e. store, upload) multiple objects using a single request, client application uses POST method described in 6.9.5 on the resource defined in 6.9. Note that the object(s) creation request contains both the Object data structure and the payload.
Retrieve object’s metadata	6.2.3	To retrieve the object’s properties such as its location, its list of attributes and flags, client application uses GET method described in 6.2.3 on the resource defined in 6.2. Note that if an object’s payload can be represented as a textual string of a reasonable length then the NMS server MAY convert a payload resource to a string and add a “TextContent” attribute that will carry this string.

Delete an object	6.2.6	To delete an object’s including its payload, client application uses DELETE method described in 6.2.6 on the resource defined in 6.2.
Manage object’s flag list	6.3.3 6.3.4	To manage the flags associated with an object, client application uses GET (retrieve, 6.3.3) and PUT (create or update, 6.3.4) methods defined for the resource in section 6.3.
Manage object’s individual flag	6.4.3 6.4.4 6.4.6	To manage an individual flag associated with an object, client application uses GET (retrieve or check flag’s existance, 6.4.3), PUT (add flag, 6.3.4) and DELETE (remove flag, 6.4.6) methods defined for the resource in section 6.4.
Retrieve the entire payload of an object	6.5.3	To retrieve the entire payload of an object at once, client application uses GET method described in 6.5.3 on the resource defined in 6.5. Note that client SHOULD obtain the location of the entire payload from the object’s payloadURL element. This operation is OPTIONAL.
Retrieve a payload part of an object	6.6.3	To retrieve an individual payload part of an object, client application uses GET method described in 6.6.3 on the resource defined in 6.6. Note that client SHOULD obtain the location of the given payload part from the object’s payloadPart element.
Search for objects	6.7.5 Appendix I	To search for objects meeting certain criteria (e.g. Date, Flag, Attribute value, keyword, etc) and retrieve information about the selected objects, client application uses POST method described in 6.7.5 on the resource defined in 6.7. To set search criteria based on Attributes and Flags, client uses Attribute names and Flag names described in Appendix I and Appendix H.
Create a folder in Network Storage	6.10.5	To create a user defined folder in the Network Storage, client application uses POST method described in 6.10.5 on the resource defined in 6.10.
Manage a folder	6.11.3 6.12.4	To retrieve information about a folder (including its location, containing objects and subfolders) or delete a folder, client application uses GET (retrieve, 6.11.3) and DELETE (remove, 6.11.6) methods defined for the resource in section 6.11. To change a folder’s name, client application uses PUT method described in 6.12.4 on the lightweight resource defined in 6.12. Note that RCS system-defined root subfolders: /Default and /RCSMessageStore MAY NOT be deleted or renamed. Attempting to delete or rename /Default and /RCSMessageStore folders SHALL result in policy errors. Additionally, root folder’s name is an empty string and MAY NOT be renamed. Attempting to rename the root folder SHALL result in a policy error.

Search for folders	6.13.5	To search for folders meeting certain criteria (e.g. creation date, root folder) and retrieve information about the selected folder(s), client application uses POST method described in 6.13.5 on the resource defined in 6.13. To set search criteria based on Attributes, client uses Attribute names described in Appendix J and Appendix K.
Copy or move objects and/or folders	6.15.5 6.16.5	To copy or move object(s) and/or folder(s) (including recursive folders' content) to a designated target folder, client application uses POST (copy, 6.15.3) and POST (move, 6.16.6) methods defined for the resource in section 6.15 and 6.16 respectively.
Subscribe to notifications	6.17.5 6.17.3	To create a subscription for notifications on NMS changes, client application uses POST method described in 6.17.5 on the resource defined in 6.17. In order to sync and receive notifications for events which took place prior to time the subscription is created; client application includes a restartToken as part of the subscription creation request (i.e. POST method described in 6.17.5). To check the list of active subscription, client application uses GET method described in 6.17.3 on the resource defined in 6.17. This operation is OPTIONAL. Note that for client applications that cannot maintain notification URL, the notification URL may be previously obtained by either executing the operation 'Establish Notification Channel for Long Polling' or, if supported, 'Establish Notification Channel for WebSockets' (section 5.1 of this profile document).
Cancel subscription for notifications	6.18.6 6.18.3	To cancel a subscription for notifications, client application uses DELETE method described in 6.18.6 on the resource defined in 6.18. To read (and for instance check the whether a given asubscription is still valid or check its expiration), client application uses GET method described in 6.18.3 on the resource defined in 6.18.
Sync up with network storage starting at restartToken	6.18.5	To sync up with network storage using an active subscription, client application uses POST method described in 6.18.5 on the resource defined in 6.18. This operation can also be used to extend the expiry time of a valid subscription.
Receive notifications about network storage changes	6.19.5	Client application receives notifications about network storage changes as described in section 6.19.5, at the notification URL provided when executing 'Subscribe to notifications' operation

Lookup the resource URL for an object or folder based on its pathname (location)	6.8.3 6.8.5 6.14.3 6.14.5	To lookup the resource URL of object(s), client application uses GET (or POST) method described in section 6.8.3 (or 6.8.5) on the resource defined in 6.8. To lookup the resource URL of folder(s), client application uses GET (or POST) method described in 6.14.3 (or 6.14.5) on the resource defined in 6.14.

5.15 WebRTC Signaling

The RCS Profile of the WebRTC Signaling API defines a subset of the HTTP resources/methods in [REST_NetAPI_WRTCSig] as listed below.

Implementations of the RCS profile MUST support the elements “allowVideoUpgrade” and “serviceType” and MUST allow the value “rcsipcall” for “serviceType”.

RCS Operations	REST Methods	Comments
Subscribe to notifications	6.1.5	To create a subscription for notifications on updates related to WebRTC signaling, client application uses POST method described in 6.1.5 on the resource defined in 6.1. Note that for client applications that cannot maintain notification URL, the notification URL SHALL be previously obtained by either executing the operation ‘Establish Notification Channel for Long Polling’ or, if supported, ‘Establish Notification Channel for WebSockets’ (section 5.1).
Cancel subscription for notifications	6.2.6	To cancel a subscription for notifications, client application uses DELETE method described in 6.2.6 on the resource defined in 6.2.
Initiate a voice or video call to a called party	6.3.5	To initiate a voice or video call to a called party, client application uses POST method described in 6.3.5 on the resource defined in 6.3.
Cancel a call setup	6.4.6	To cancel a call set-up, client application uses DELETE method described in 6.4.6 on the resource defined in 6.4.
Receive notifications about unsuccessful call setup, such as “busy”, “not reachable”, “no answer” and ”declined”	6.10.5	Client application receives notifications about unsuccessful call setup as described in section 6.10.5, at the notification URL provided when executing ‘Subscribe to notifications’ operation
Receive notifications about successful call setup	6.12.5	Client application receives notifications about successful call setup as described in section 6.12.5, at the notification URL provided when executing ‘Subscribe to notifications’ operation
Receive notifications about a new incoming voice or video call.	6.11.5	Client application receives notifications about incoming voice or video call as described in section 6.11.5, at the notification URL provided when executing ‘Subscribe to notifications’ operation
Accept an incoming voice or video call	6.5.4	To accept an incoming call, client application uses PUT method described in 6.5.4 on the resource defined in 6.5.

Reject an incoming voice or video call	6.4.6	To reject an incoming call, client application uses DELETE method described in 6.4.6 on the resource defined in 6.4.
Terminate a voice or video call	6.4.6	To terminate an ongoing call, client application uses DELETE method described in 6.4.6 on the resource defined in 6.4.
Request upgrading a call from voice to video	6.8.4	To request upgrading a call from voice to video, client application uses PUT method described in 6.8.4 on the resource defined in 6.8, providing an update offer that declares additional video media descriptions.
Receive notification requesting upgrading a call from voice to video	6.13.5	Client application receives notifications about an update offer requesting an upgrade as described in section 6.13.5, at the notification URL provided when executing ‘Subscribe to notifications’ operation
Accept upgrading a call from voice to video	6.7.4	To accept an offer to upgrade a call from voice to video, client application uses PUT method described in 6.7.4 on the resource defined in 6.7, providing an answer to the update offer, declaring its own video media descriptions, including the fact whether or not it intends to send back video.
Reject upgrading a call from voice to video	6.8.6	To reject an offer to upgrade a call from voice to video, client application uses DELETE method described in 6.8.6 on the resource defined in 6.8, removing to update offer.
Receive notification about acceptance of an upgrade request	6.12.5	Client application receives notifications about acceptance of an update request as described in section 6.12.5, at the notification URL provided when executing ‘Subscribe to notifications’ operation
Receive notification about rejection of an upgrade request	6.10.5	Client application receives a notification indicating rejection of an update request as described in section 6.10.5, at the notification URL provided when executing ‘Subscribe to notifications’ operation. This is signaled using the event type ‘Declined’ in the notification.
Request downgrading a call from video to voice	6.8.4	To request downgrading a call from video to voice, client application uses PUT method described in 6.8.4 on the resource defined in 6.8, providing an update offer that removes the video media descriptions from the session description.
Receive notification about downgrading a call from video to voice	6.13.5	Client application receives notifications about an update offer requesting a downgrade as described in section 6.13.5, at the notification URL provided when executing ‘Subscribe to notifications’ operation. Note that the downgrade from a video call to a voice call does not require the remote user to accept it, nevertheless the application which receives this notification is expected to provide an answer.

Provide an answer to an update offer	6.7.4	To provide an answer to an update offer (which usually changes some media parameters, but may not necessarily mean a session upgrade or downgrade), client application uses PUT method described in 6.7.4 on the resource defined in 6.7, providing an answer to the update offer. This operations needs to be supported in addition to the ones required by [REQ_RCS].
Receive notification about an answer	6.14.5	Client application receives notifications about an answer as described in section 6.13.5, at the notification URL provided when executing 'Subscribe to notifications' operation. Note that an answer may be transmitted in an early phase during call setup, to create a media path when the remote user has not yet accepted the call, or even prior to the remote user being alerted. Such schemes are known as "early media". This operations needs to be supported in addition to the ones required by [REQ_RCS].
Update ICE status	6.9.4	To update the ICE status, client application uses PUT method described in 6.9.4 on the resource defined in 6.9. This operations needs to be supported in addition to the ones required by [REQ_RCS] if the client runs in a web browser.
Receive notification about conflicts	6.16.5	Client application receives notifications about a conflict as described in section 6.16.5, at the notification URL provided when executing 'Subscribe to notifications' operation. This operations needs to be supported in addition to the ones required by [REQ_RCS].
Receive notification about subscription cancellation	6.15.5	Client application receives notifications about the cancellation of its subscription as described in section 6.15.5, at the notification URL provided when executing 'Subscribe to notifications' operation. This operation should be supported in addition to the ones required by [REQ_RCS], as the client needs to renew its subscription in this case.

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
n/a	n/a	No prior version

A.2 Draft Version 3.0 History

Document Identifier	Date	Sections	Description
Draft Versions OMA-TS-REST_NetAPI_RCSProfile-V3_0	19 Mar 2013	all	First draft baseline
	29 Apr 2013	all	Incorporated OMA-ARC-RCS-Profile3-2013-0001-CR_RCS3_update_accept_by_PUT
	18 Jun 2013	5.4	Incorporated OMA-ARC-RCS-Profile3-2013-0003-CR_Call_API_fix_RCS_Profile
	18 Jul 2013	all	Editorial updates
	17 Jan 2014	all	Editorial updates and Incorporated OMA-ARC-RCS-Profile3-2014-0004-CR_Adding_WRTCSig
	31 Jan 2014	all	Incorporated OMA-ARC-RCS-Profile3-2014-0005R02-CR_Adding_Ping_Pong_to_RCS3_Profile
	01 Sep 2014	5.13, 5.14, B.13, B.14	Incorporated CRs: OMA-ARC-RCS-Profile3-2014-0007R01-CR_Filling_NMS_section OMA-ARC-RCS-Profile3-2014-0008R01-CR_TS_Adding_missing_information_for_Presence
	20 Oct 2014	5.13	Incorporated CR: OMA-ARC-RCS-Profile3-2014-0009-CR_CONR_Cap_Disc_Presence_Lists
	18 Dec 2014	Many	Incorporated CRs: OMA-ARC-RCS-Profile3-2014-0013-CR_CONR_comments_A002_to_A005 OMA-ARC-RCS-Profile3-2014-0014R02-CR_TS_CONRR_editorial_comments_resolution OMA-ARC-RCS-Profile3-2014-0017R01-CR_solution_to_resolve_comments_related_to_Long_lived_group_chat OMA-ARC-RCS-Profile3-2014-0019R01-CR_CONR_minor_markups OMA-ARC-RCS-Profile3-2014-0020R01-CR_CONR_NMS_notification OMA-ARC-RCS-Profile3-2014-0022-CR_CONR_Group_Chat_SCR OMA-ARC-RCS-Profile3-2014-0024R01-CR_TS_adding_information_for_Messaging OMA-ARC-RCS-Profile3-2014-0023R01-CR_CONRR_non_solved_issues_as_restrictions
Candidate Version OMA-TS-REST_NetAPI_RCSProfile-V3_0	03 Feb 2015	n/a	Status changed to Candidate by TP TP Ref # OMA-TP-2015-0034-INP_REST_NetAPI_RCSProfile_V3_0_ERP_and_ETR_for_Candidate_approval

Appendix B. Static Conformance Requirements (Normative)

The notation used in this appendix is specified in [SCRRULES].

B.1 SCR for RCS.NotificationChannel Server

Support for the RCSProfile of the RESTful Network API for Notification Channel implies supporting the following SCRs as defined in the SCR tables of [REST_NetAPI_NotifChnl].

Item	Function	Reference	Requirement
RCS-NOTIFCHNL-S-001-O	Support for the RESTful Notification Channel API	[REST_NetAPI_NotifChnl] Appendix B	REST-NC-SUPPORT-S-001-M AND REST-NC-SUPPORT-S-002-M AND REST-NC-SUPPORT-S-003-M AND REST-NC-CHANNELS-S-001-M AND REST-NC-CHANNELS-S-003-O AND REST-NC-INDCHANNEL-S-001-M AND REST-NC-INDCHANNEL-S-002-M AND REST-NC-INDCHANNEL-S-003-M AND REST-NC-LONGPOLL-S-001-O AND REST-NC-LONGPOLL-S-002-O

B.2 SCR for RCS.Chat Server

Support for the RCS Profile of the RESTful Network API for Chat implies supporting the following SCRs defined in the table below.

Item	Function	Reference	Requirement
RCS-CHAT-S-001-O	Support for the RESTful Chat API	[REST_NetAPI_Chat] Appendix B	REST-CHAT-SUPPORT-S-001-M AND REST- CHAT-SUPPORT-S-002-M AND REST-CHAT-SUPPORT-S-003-M AND REST-CHAT-SUBSCR- S-001-M AND REST-CHAT- SUBSCR-S-003-M AND REST-CHAT-INDSUBSCR -S-001-M AND REST-CHAT-INDSUBSCR-S-003-M AND REST-CHAT-ONE2ONE-SESS-S-001-M

Item	Function	Reference	Requirement
			AND REST-CHAT-ONE2ONE-SESS-S-002-O AND REST-CHAT-ONE2ONE-INDSESS-CONF-S-001-O AND REST-CHAT-ONE2ONE-INDSESS-CONF-S-003-O AND REST-CHAT-ONE2ONE-INDSESS-ADH-S-001-O AND REST-CHAT-ONE2ONE-INDSESS-STAT-S-001-O AND REST-CHAT-ONE2ONE-INDSESS-STAT-S-002-O AND REST-CHAT-ONE2ONE-INDSESS-EXT-S-001-O AND REST-CHAT-ONE2ONE-INDSESS-EXT-S-002-O AND REST-CHAT-ONE2ONE-MSG-S-001-M AND REST-CHAT-ONE2ONE-MSG-S-002-M AND REST-CHAT-ONE2ONE-INDMSG-STAT-S-001-M AND REST-CHAT-ONE2ONE-INDMSG-STAT-S-003-M AND REST-CHAT-GROUP-SESS-S-001-M AND REST-CHAT-GROUP-SESS-S-002-M AND REST-CHAT-GROUP-SESS-S-003-M AND REST-CHAT-GROUP-INDSESS-S-001-M AND REST-CHAT-GROUP-INDSESS-S-003-O AND REST-CHAT-GROUP-INDSESS-S-004-M AND REST-CHAT-GROUP-INDSESS-PART-S-001-M AND REST-CHAT-GROUP-INDSESS-PART-S-003-M AND REST-CHAT-GROUP-INDSESS-INDPART-S-001-M AND REST-CHAT-GROUP-INDSESS-INDPART-S-003-M

Item	Function	Reference	Requirement
			AND REST-CHAT-GROUP-INDSESS-INDPART-S-001-M AND REST-CHAT-GROUP-INDSESS-INDPART-S-003-M AND REST-CHAT-GROUP-MSG-S-001-M AND REST-CHAT-GROUP-MSG-S-002-M AND REST-CHAT-NOTIF MSG-S-001-M AND REST-CHAT-NOTIF MSG-S-002-M AND REST-CHAT-NOTIF-MSG-STAT-S-001-M AND REST-CHAT-NOTIF-MSG-STAT-S-002-M AND REST-CHAT-NOTIF-ONE2ONE-INVITE-S-001-O AND REST-CHAT-NOTIF-ONE2ONE-INVITE-S-002-O AND REST-CHAT-NOTIF-GROUP-INVITE-S-001-M AND REST-CHAT-NOTIF-GROUP-INVITE-S-002-M AND REST-CHAT-NOTIF-EVENT-S-001-M AND REST-CHAT-NOTIF-EVENT-S-002-M AND REST-CHAT-NOTIF-GROUP-PART-S-001-M AND REST-CHAT-NOTIF-GROUP-PART-S-002-M

B.3 SCR for RCS.FileTransfer Server

Support for the RCS Profile of the RESTful Network API for File Transfer implies supporting all mandatory SCRs as defined in the SCR tables of [REST_NetAPI_FileTransfer].

Item	Function	Reference	Requirement
RCS-FILETRANSFER-S-001-O	Support for the RESTful File Transfer API	[REST_NetAPI_FileTransfer] Appendix B	REST-FileTransfer-SUPPORT-S-001-M AND REST-FileTransfer-SUPPORT-S-002-M AND REST-FileTransfer-SUPPORT-S-003-M AND REST-FileTransfer- SUBSCR-001-M

Item	Function	Reference	Requirement
			AND REST-FileTransfer- SUBSCR-003-M AND REST-FileTransfer- IND-SUBSCR-001-M AND REST-FileTransfer- IND-SUBSCR-003-M AND REST-FileTransfer-Sess-001-M AND REST-FileTransfer-Sess-002-M AND REST-FileTransfer-IND-Sess-001-M AND REST-FileTransfer-IND-Sess-003-M AND REST-FileTransfer-IND-Sess-Stat-001-M AND REST-FileTransfer-IND-Sess-Stat-002-M AND REST-FileTransfer-INV-NOTIF-001-M AND REST-FileTransfer-INV-NOTIF-002-M AND REST-FileTransfer-Event-NOTIF-001-M AND REST-FileTransfer-Event-NOTIF-002-M AND REST-FileTransfer-Link-NOTIF-001-M AND REST-FileTransfer-Link-NOTIF-002-M AND REST-FileTransfer-RA-NOTIF-001-M AND REST-FileTransfer-RA-NOTIF-002-M

B.4 SCR for RCS.ThirdPartyCall Server

Support for the RCS Profile of the RESTful Network API for Third Party Call implies supporting all mandatory SCRs as defined in the SCR tables of [REST_NetAPI_3PC].

Item	Function	Reference	Requirement
RCS-3PC-S-001-O	Support for the RESTful ThirdParty Call API	[REST_NetAPI_3PC] Appendix B	REST-3PC-SUPPORT-S-001-M AND REST-3PC-SUPPORT-S-002-M AND REST-3PC-SUPPORT-S-003-M AND REST-3PC-SESS-S-001-M

Item	Function	Reference	Requirement
			AND REST-3PC-SESS-S-003-M AND REST-3PC-SESS-S-005-M AND REST-3PC-INDSESS-S-001-M AND REST-3PC-INDSESS-S-002-M AND REST-3PC-INDSESS-S-003-M AND REST-3PC-INDSESS-PART-S-001-M AND REST-3PC-INDSESS-PART-S-002-M AND REST-3PC-INDSESS-PART-S-003-M AND REST-3PC-INDSESS-INDPART-S-001-M AND REST-3PC-INDSESS-INDPART-S-002-M AND REST-3PC-INDSESS-INDPART-S-003-M

B.5 SCR for RCS.CallNotification Server

Support for the RCS Profile of the RESTful Network API for Call Notification implies supporting all mandatory SCRs as defined in the SCR tables of [REST_NetAPI_CallNotif].

Item	Function	Reference	Requirement
RCS-CALLNOTIF-S-001-O	Support for the RESTful Call Notification API	[REST_NetAPI_CallNotif] Appendix B	REST-CN-SUPPORT-S-001-M AND REST-CN-SUPPORT-S-002-M AND REST-CN-SUPPORT-S-003-M AND REST-CN-SUBSCR-CALLEVENT-S-001-M AND REST-CN- SUBSCR-CALLEVENT-S-003-M AND REST-CN-SUBSCR-INDCALLEVENT-S-001-M AND REST-CN-SUBSCR-INDCALLEVENT-S-002-M AND REST-CN-SUBSCR-INDCALLEVENT-S-003-M AND REST-CN-SUBSCR-PAC-S-001-M AND REST-CN-SUBSCR-PAC-S-003-M AND

Item	Function	Reference	Requirement
			REST-CN-SUBSCR-INDPAC-S-001-M AND REST-CN-SUBSCR-INDPAC-S-002-M AND REST-CN-SUBSCR-INDPAC-S-003-M AND REST-CN-NOTIF-CALLEVENT-S-001-M AND REST-CN-NOTIF-CALLEVENT-S-002-M AND REST-CN-NOTIF-MEDINT-S-001-M AND REST-CN- NOTIF-MEDINT-S-002-M

B.6 SCR for RCS.VideoShare Server

Support for the RCS Profile of the RESTful Network API for Video Share implies supporting all mandatory SCRs as defined in the SCR tables of [REST_NetAPI_VideoShare].

Item	Function	Reference	Requirement
RCS-VIDEOSHARE-S-001-O	Support for the RESTful Video Share API	[REST_NetAPI_VideoShare] Appendix B	REST-VIDEOSHARE-SUPPORT-S-001-M AND REST-VIDEOSHARE-SUPPORT-S-002-M AND REST-VIDEOSHARE-SUPPORT-S-003-M AND REST-VIDEOSHARE-SUBSCR-S-001-M AND REST-VIDEOSHARE- SUBSCR-S-003-M AND REST-VIDEOSHARE-IND-SUBSCR-S-001-M AND REST-VIDEOSHARE-IND-SUBSCR-S-003-M AND REST-VIDEOSHARE-SESS-S-001-M AND REST-VIDEOSHARE-SESS-S-002-M. AND REST-VIDEOSHARE-IND-SESS-S-001-M AND REST-VIDEOSHARE-IND-SESS-S-003-M AND REST-VIDEOSHARE-IND-SESS-STAT-S-001-M AND REST-VIDEOSHARE-IND-SESS-STAT-S-002-M AND REST-VIDEOSHARE-INVITE-NOTIF-S-001-M AND REST-VIDEOSHARE-INVITE-NOTIF-S-002-M

Item	Function	Reference	Requirement
			AND REST-VIDEOSHARE-ACCEPT-NOTIF-S-001-M AND REST-VIDEOSHARE-ACCEPT-NOTIF-S-002-M AND REST-VIDEOSHARE-EVENT-NOTIF-S-001-M AND REST-VIDEOSHARE-EVENT-NOTIF-S-002-M

B.7 SCR for RCS.ImageShare Server

Support for the RCS Profile of the RESTful Network API for Image Share implies supporting all mandatory SCRs as defined in the SCR tables of [REST_NetAPI_ImageShare].

Item	Function	Reference	Requirement
RCS-IMAGESHARE-S-001-O	Support for the RESTful Image Share API	[REST_NetAPI_ImageShare] Appendix B	REST-IMAGESHARE-SUPPORT-S-001-M AND REST-IMAGESHARE-SUPPORT-S-002-M AND REST-IMAGESHARE-SUPPORT-S-003-M AND REST-IMAGESHARE-SUBSCR-S-001-M AND REST-IMAGESHARE- SUBSCR-S-003-M AND REST-IMAGESHARE-IND-SUBSCR-S-001-M AND REST-IMAGESHARE-IND-SUBSCR-S-003-M AND REST-IMAGESHARE-SESS-S-001-M AND REST-IMAGESHARE-SESS-S-002-M. AND REST-IMAGESHARE-IND-SESS-S-001-M AND REST-IMAGESHARE-IND-SESS-S-003-M AND REST-IMAGESHARE-IND-SESS-STAT-S-001-M AND REST-IMAGESHARE-IND-SESS-STAT-S-002-M AND REST-IMAGESHARE-INVITE-NOTIF-S-001-M AND REST-IMAGESHARE-INVITE-NOTIF-S-002-M AND REST-IMAGESHARE-ACCEPT-NOTIF-S-001-M AND REST-IMAGESHARE-ACCEPT-NOTIF-S-002-M AND

Item	Function	Reference	Requirement
			REST-IMAGESHARE-LINK-NOTIF-S-001-M AND REST-IMAGESHARE-LINK-NOTIF-S-002-M AND REST-IMAGESHARE-EVENT-NOTIF-S-001-M AND REST-IMAGESHARE-EVENT-NOTIF-S-002-M

B.8 SCR for RCS.Messaging Server

Support for the RCS Profile of the RESTful Network API for Messaging implies supporting all mandatory SCRs as defined in the SCR tables of [REST_NetAPI_Messaging].

Item	Function	Reference	Requirement
RCS-MESSAGING-S-001-O	Support for the RESTful Messaging API	[REST_NetAPI_Messaging] Appendix B	REST-MSG-SUPPORT-S-001-M AND REST-MSG-SUPPORT-S-002-M AND REST-MSG-SUPPORT-S-003-M AND REST-MSG-OUTB-S-001-M AND REST-MSG-OUTB-S-003-M AND REST-MSG-INB-ONL-SUBSCR-S-001-M AND REST-MSG-INB-ONL-SUBSCR-S-003-M AND REST-MSG-INB-INDON-SUBSCR-S-003-M AND REST-MSG-INB-NOTIF-S-001-M AND REST-MSG-INB-NOTIF-S-002-M AND REST-MSG-OUTB-DELSTAT-NOTIF-S-001-M AND REST-MSG-OUTB-DELSTAT-NOTIF-S-002-M AND REST-MSG-IND-INB-STAT-S-002-M

B.9 SCR for RCS.NetworkAddressBook Server

Support for the RCS Profile of the RESTful Network API for Address Book implies supporting all mandatory SCRs as defined in the SCR tables of [REST_NetAPI_AddressBook].

Item	Function	Reference	Requirement
RCS-NETAB-S-001-O	Support for the RESTful AddressBook API	[REST_NetAPI_AddressBook] Appendix B	REST-AB-SUPPORT-S-001-M AND REST-AB-SUPPORT-S-002-M AND

Item	Function	Reference	Requirement
			REST-AB-SUPPORT-S-003-M AND REST-AB-CONTACT-COL-S-001-M AND REST-AB-CONTACT-COL-S-002-M AND REST-AB-CONTACT-IND-S-001-M AND REST-AB-CONTACT-IND-S-002-M AND REST-AB-CONTACT-IND-S-003-M AND REST-AB-CONTACT-IND-S-004-M AND REST-AB-ATTRIB-CONTACT-S-001-M AND REST-AB-ATTRIB-CONTACT-S-002-M AND REST-AB-IND-ATTRIB-CONTACT-S-001-M AND REST-AB-IND-ATTRIB-CONTACT-S-002-M AND REST-AB-IND-ATTRIB-CONTACT-S-003-M AND REST-AB-IND-ATTRIB-CONTACT-S-004-M AND REST-AB-SUBSCR-S-001-M AND REST-AB-SUBSCR-S-002-M AND REST-AB-SUBSCR-S-003-M AND REST-AB-IND-SUBSCR-S-001-M AND REST-AB-IND-SUBSCR-S-002-M AND REST-AB-IND-SUBSCR-S-003-M AND REST-AB-IND-SUBSCR-S-004-M AND REST-AB-NOTIF-S-001-M AND REST-AB-NOTIF-S-002-M AND REST-AB-IND-MEM-TRANS-S-001-M AND REST-AB-IND-MEM-TRANS-S-002-M

B.10 SCR for RCS.ACR Server

Support for the RCS Profile of the RESTful Network API for Anonymous Customer Reference Management implies supporting all mandatory SCRs as defined in the SCR tables of [REST_NetAPI_ACR].

Item	Function	Reference	Requirement
RCS-ACRM-S-001-O	Support for the RESTful Anonymous Customer Reference Management API	[REST_NetAPI_ACR] Appendix B	REST-ACRMGNT-SUPPORT-S-001-M AND REST-ACRMGNT-SUPPORT-S-002-M AND REST-ACRMGNT-SUPPORT-S-003-M AND REST-ACRMGNT-USERSAPP-S-001-M AND REST-ACRMGNT-USERSAPP-S-002-M AND REST-ACRMGNT-USERSAPP-S-003-M AND REST-ACRMGNT-ACR-S-001-M AND REST-ACRMGNT-ACR-S-002-M AND REST-ACRMGNT-ACR-S-003-M AND REST-ACRMGNT-ACRSTATUS-S-001-M AND REST-ACRMGNT-ACRSTATUS-S-002-M AND REST-ACRMGNT-ACRSTATUS-S-003-M

B.11 SCR for RCS.CapabilityDiscovery Server

Support for the RCS Profile of the RESTful Network API for Capability Discovery implies supporting all mandatory SCRs as defined in the SCR tables of [REST_NetAPI_CapabilityDiscovery].

Item	Function	Reference	Requirement
RCS-CAPDIS-S-001-O	Support for the RESTful CapabilityDiscovery API	[REST_NetAPI_ CapabilityDiscovery] Appendix B	REST-CAPDIS-SUPPORT-S-001-M AND REST-CAPDIS-SUPPORT-S-002-M AND REST-CAPDIS-SUPPORT-S-003-M AND REST-CAPDIS-OWN-CAPSRC-S-001-M AND REST-CAPDIS-OWN-CAPSRC-S-002-M AND REST-CAPDIS-OWN-CAPSRC-S-003-M AND REST-CAPDIS-OWN-CAPS-S-001-M AND REST-CAPDIS-OWN-CAPS-S-002-M

Item	Function	Reference	Requirement
			AND REST-CAPDIS-OWN-CAPS-S-003-M AND REST-CAPDIS-OWN-CAPS-S-004-M AND REST-CAPDIS-OWN-IND-CAP-S-001-O AND REST-CAPDIS-OWN-IND-CAP-S-002-O AND REST-CAPDIS-OWN-IND-CAP-S-003-O AND REST-CAPDIS-OWN-IND-CAP-S-004-O AND REST-CAPDIS-CONTACT-CAPS-S-001-M AND REST-CAPDIS-CONTACT-CAPS-S-002-M AND REST-CAPDIS-CONTACT-CAPS-S-004-M

B.12 SCR for RCS.TerminalLocation Server

Support for the RCS Profile of the RESTful Network API for Terminal Location implies supporting all mandatory SCRs as defined in the SCR tables of [REST_NetAPI_Location].

Item	Function	Reference	Requirement
RCS-LOCATION-S-001-O	Support for the RESTful Terminal Location API	[REST_NetAPI_Location] Appendix B	REST-LOC-SUPPORT-S-001-M AND REST-LOC-SUPPORT-S-002-M AND REST-LOC-SUPPORT-S-003-M AND REST-LOC-LOC-S-001-M AND REST-LOC-LOC-S-002-M

B.13 SCR for RCS.Presence Server

Support for the RCS Profile of the RESTful Network API for Presence implies supporting all mandatory SCRs as defined in the SCR tables of [REST_NetAPI_Presence].

Item	Function	Reference	Requirement
RCS-PRESENCE-S-001-O	Support for the RESTful Presence API	[REST_NetAPI_Presence] Appendix B	REST-PRESENCE-SUPPORT-S-001-M AND REST-PRESENCE-SUPPORT-S-002-M AND REST-PRESENCE-SUPPORT-S-003-M AND REST-PRESENCE-PRES-PS-S-001-M AND

Item	Function	Reference	Requirement
			REST-PRESENCE-PRES-PS-S-002-M AND REST-PRESENCE-PRES-PS-S-003-M AND REST-PRESENCE-PRES-IND-PS-S-001-M AND REST-PRESENCE-PRES-IND-PS-S-002-M AND REST-PRESENCE-PRES-IND-PS-S-003-M AND REST-PRESENCE-PRES-IND-PS-S-004-M AND REST-PRESENCE-PRES-PRSOURCE-PERS-S-001-M AND REST-PRESENCE-PRES-PS-PERS-S-002-M AND REST-PRESENCE-PRES-PS-PERS-S-003-M AND REST-PRESENCE-PRES-PS-PERS-S-004-M AND REST-PRESENCE-PRES-CONTL-S-001-M AND REST-PRESENCE-PRES-CONTL-S-002-M AND REST-PRESENCE-PRES-IND-CONT-S-001-M AND REST-PRESENCE-PRES-IND-CONT-S-002-M AND REST-PRESENCE-PRES-IND-CONT-S-003-M AND REST-PRESENCE-PRES-IND-CONT-S-004-M AND REST-PRESENCE-PRES-WL-S-001-M AND REST-PRESENCE-PRES-WL-S-002-M AND REST-PRESENCE-PRES-IND-WATCHER-S-001-M AND REST-PRESENCE-PRES-IND-WATCHER-S-002-M AND REST-PRESENCE-PRES-AUTH-RULES-S-001-M AND REST-PRESENCE-PRES-AUTH-RULES-S-002-M AND REST-PRESENCE-PRES-AUTH-RULES-S-003-M

Item	Function	Reference	Requirement
			AND REST-PRESENCE-PRES-IND-AUTH-RULE-S-001-M AND REST-PRESENCE-PRES-IND-AUTH-RULE-S-002-M AND REST-PRESENCE-PRES-IND-AUTH-RULE-S-003-M AND REST-PRESENCE-PRES-IND-AUTH-RULE-S-004-M AND REST-PRESENCE-WATCH-PC-S-001-M AND REST-PRESENCE-WATCH-PC-S-002-M AND REST-PRESENCE-WATCH-PL-S-001-M AND REST-PRESENCE-WATCH-PL-S-002-M AND REST-PRESENCE-WATCH-PCC-S-001-M AND REST-PRESENCE-WATCH-PCC-S-002-M AND REST-PRESENCE-PRES-SUBSCR-WS-S-001-M AND REST-PRESENCE-PRES-SUBSCR-WS-S-003-M AND REST-PRESENCE-PRES-IND-SUBSCR-WS-S-001-M AND REST-PRESENCE-PRES-IND-SUBSCR-WS-S-003-M AND REST-PRESENCE-PRES-IND-SUBSCR-WS-S-004-M AND REST-PRESENCE-WS-NOTIF-S-001-M AND REST-PRESENCE-PRES-WS-S-002-M AND REST-PRESENCE-WATCH-SUBSCR-PS-SINGP-S-001-M AND REST-PRESENCE-WATCH-SUBSCR-PS-SINGP-S-003-M AND REST-PRESENCE-WATCH-IND-SUBSCR-PS-

Item	Function	Reference	Requirement
			SINGP-S-001-M AND REST-PRESENCE-WATCH-IND-SUBSCR-PS-SINGP-S-003-M AND REST-PRESENCE-WATCH-IND-SUBSCR-PS-SINGP-S-004-M AND REST-PRESENCE-PS-NOTIF-S-001-M AND REST-PRESENCE-PS-NOTIF-S-002-M AND REST-PRESENCE-WATCH-SUBSCR-PLS-SINGPL-S-001-M AND REST-PRESENCE-WATCH-SUBSCR-PLS-SINGPL-S-003-M AND REST-PRESENCE-WATCH-IND-SUBSCR-PLS-SINGPL-S-001-M AND REST-PRESENCE-WATCH-IND-SUBSCR-PLS-SINGPL-S-003-M AND REST-PRESENCE-WATCH-IND-SUBSCR-PLS-SINGPL-S-004-M AND REST-PRESENCE-PLS-NOTIF-S-001-M AND REST-PRESENCE-PLS-NOTIF-S-002-M AND REST-PRESENCE-PRES-PORTR-ICON-S-001-M AND REST-PRESENCE-PRES-PORTR-ICON-S-002-M AND REST-PRESENCE-PRES-PORTR-ICON-S-003-M AND REST-PRESENCE-PRES-PORTR-ICON-S-004-M AND REST-PRESENCE-WATCH-APL-S-001-O AND REST-PRESENCE-WATCH-APL-S-002-O

B.14 SCR for RCS.NMS Server

Support for the RCS Profile of the RESTful Network API for NMS implies supporting the following SCRs as defined in the SCR tables of [REST_NetAPI_NMS].

Item	Function	Reference	Requirement
RCS-NMS-S-001-	Support for the	[REST_NetAPI_NMS]	REST-NMS-SUPPORT-S-001-M

Item	Function	Reference	Requirement
O	RESTful NMS API	Appendix B	AND REST- NMS-SUPPORT-S-002-M AND REST- NMS-SUPPORT-S-003-M AND REST-NMS-OBJECTS-S-001-M AND REST-NMS-OBJECTS-S-002-M AND REST-NMS-AOBJECT-S-001-M AND REST-NMS-AOBJECT-S-002-M AND REST-NMS-AOBJECT-S-003-M AND REST-NMS-AOBJECT-FLAGS-S-001-M AND REST-NMS-AOBJECT-FLAGS-S-002-M AND REST-NMS-AOBJECT-FLAGS-S-004-M AND REST-NMS-AOBJECT-INDFLAG-S-001-M AND REST-NMS-AOBJECT-INDFLAG-S-002-M AND REST-NMS-AOBJECT-INDFLAG-S-003-M AND REST-NMS-AOBJECT-INDFLAG-S-004-M AND REST-NMS-AOBJECT-PAYLOAD-S-001-M AND REST-NMS-AOBJECT-PAYLOAD-S-002-M AND REST-NMS-AOBJECT-PAYLOADPART-S-001-M AND REST-NMS-AOBJECT-PAYLOADPART-S-002-M AND REST-NMS-OBJECTS-SEARCH-S-001-M AND REST-NMS-OBJECTS-SEARCH-S-002-M AND REST-NMS-OBJECTS-PATHTOID-S-001-M AND REST-NMS-OBJECTS-PATHTOID-S-001-M AND REST-NMS-OBJECTS-PATHTOID-S-002-M

Item	Function	Reference	Requirement
			AND REST-NMS-FOLDERS-S-001-M AND REST-NMS-FOLDERS-S-002-M AND REST-NMS-AFOLDERS-S-001-M AND REST-NMS-AFOLDERS-S-002-M AND REST-NMS-AFOLDERS-S-003-M AND REST-NMS-FOLDERNAME-S-001-M AND REST-NMS-FOLDERNAME-S-002-M AND REST-NMS-FOLDERNAME-S-003-M AND REST-NMS-FOLDERS-SEARCH-S-001-M AND REST-NMS-FOLDERS-SEARCH-S-002-M AND REST-NMS-FOLDERS-PATHTOID-S-001-M AND REST-NMS-FOLDERS-PATHTOID-S-002-M AND REST-NMS-FOLDERS-PATHTOID-S-003-M AND REST-NMS-FOLDERS-COPY-S-001-M AND REST-NMS-FOLDERS-COPY-S-002-M AND REST-NMS-FOLDERS-COPY-S-003-M AND REST-NMS-FOLDERS-MOVE-S-001-M AND REST-NMS-FOLDERS-MOVE-S-002-M AND REST-NMS-FOLDERS-MOVE-S-003-M AND REST-NMS-SUBSCR-S-001-O AND REST-NMS-SUBSCR-S-003-O AND REST-NMS-SUBSCR-S-004-O AND REST-NMS-SUBSCR-S-005-O AND REST-NMS-INDSUBSCR-S-001-O

Item	Function	Reference	Requirement
			AND REST-NMS-INDSUBSCR-S-002-O AND REST-NMS-INDSUBSCR-S-003-O AND REST-NMS-INDSUBSCR-S-004-O AND REST-NMS-NOTIF-S-001-O AND REST-NMS-NOTIF-S-002-O

B.15 SCR for RCS.WRTCSig Server

Support for the RCS Profile of the RESTful Network API for WebRTC Signaling implies supporting the following SCRs as defined in the SCR tables of [REST_NetAPI_WRTCSig].

Item	Function	Reference	Requirement
RCS-WRTCSIG-S-001-O	Support for the RESTful WebRTC Signaling API	[REST_NetAPI_WRTCSig] Appendix B	REST-WRTCSIG-SUPPORT-S-001-M AND REST- WRTCSIG-SUPPORT-S-002-M AND REST- WRTCSIG-SUPPORT-S-003-M AND REST-WRTCSIG-SUBSCR-S-001-M AND REST-WRTCSIG-SUBSCR-S-003-M AND REST-WRTCSIG-INDSUBSCR-S-001-M AND REST-WRTCSIG-INDSUBSCR-S-003-M AND REST-WRTCSIG-SESS-S-001-M AND REST-WRTCSIG-SESS-S-002-M AND REST-WRTCSIG-INDSESS-S-001-M AND REST-WRTCSIG-INDSESS-S-003-M AND REST-WRTCSIG-INDSESS-S-004-M AND REST-WRTCSIG-INDSESS-S-005-M AND REST-WRTCSIG-STATUS-S-001-M AND REST-WRTCSIG-STATUS-S-003-M AND REST-WRTCSIG-ANSWER-S-001-M AND

Item	Function	Reference	Requirement
			REST-WRTCSIG-ANSWER-S-003-M AND REST-WRTCSIG-UPDATE-S-001-M AND REST-WRTCSIG- UPDATE-S-003-M AND REST-WRTCSIG- UPDATE-S-004-M AND REST-WRTCSIG-ICESTAT-S-001-M AND REST-WRTCSIG-ICESTAT-S-003-M AND REST-WRTCSIG-NOTIF-EVENT-S-001-M AND REST-WRTCSIG-NOTIF-EVENT-S-002-M AND REST-WRTCSIG-NOTIF-INVITE-S-001-M AND REST-WRTCSIG-NOTIF-INVITE-S-002-M AND REST-WRTCSIG-NOTIF-ACCEPT-S-001-M AND REST-WRTCSIG-NOTIF-ACCEPT-S-002-M AND REST-WRTCSIG-NOTIF-OFFER-S-001-M AND REST-WRTCSIG-NOTIF-OFFER-S-002-M AND REST-WRTCSIG-NOTIF-ANSWER-S-001-M AND REST-WRTCSIG-NOTIF-ANSWER-S-002-M AND REST-WRTCSIG-NOTIF-SUBCXL-S-001-M AND REST-WRTCSIG-NOTIF-SUBCXL-S-002-M AND REST-WRTCSIG-NOTIF-CONFLICT-S-001-M AND REST-WRTCSIG-NOTIF-CONFLICT-S-002-M

Appendix C. Using OMA Authorization Framework for Network APIs

All RESTful Network API in this profile MAY support the authorization framework defined in [Autho4API_10]. For details on supported scope values for a specific RESTful Network API see Appendix G of that RESTful Network API (e.g. for scope values supported by [REST_NetAPI_Chat] see [REST_NetAPI_Chat] Appendix G).