Rich Communication Ecosystem

RCS-e Network API Detailed Requirements

1.0

October 17th, 2011

This is a non binding permanent reference document of the GSM Association.

Security Classification – NON CONFIDENTIAL GSMA MATERIAL

.

Table of contents

1. INTRODUCTION

Rich Communication Suit (RCS) is a new and innovative service that merges existing features trying to improve the end-user communications. RCS is a cross-industry initiative where every participant collaborates to produce this common vision of the rich communication services.

Several RCS R2 trials have been carried out by different operators in order to test the maturity of the technology and to analyze the market opportunity. The results of those trials and other customer trials provided a better understanding of where operators can further enhance their data network offering to deliver more value to customers and complement established 3rd party services. It implied a renewed specification that shall enhance some features of the current RCS R2 specification focused on new communication format called RCS-e Advanced Communications.

RCS-e does not detail the *"social presence information"* functionality described in the RCS R2 specification. However, an operator can decide to launch RCS-e service including both the RCS social presence information defined in RCS R2 specification in addition to the advanced communications services defined in RCS-e.

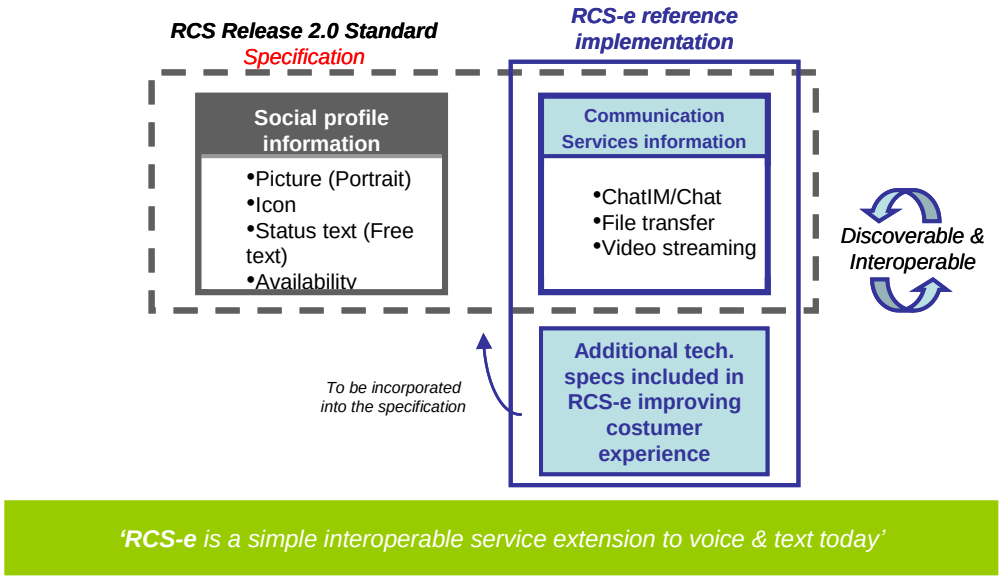

Figure 1 : RCS-e Positioning

At first glance probably there is no too much difference between RCS R2 and RCS-e versions. However, in the technical realization there are some important changes to be done. The fundamental that enables RCS-e is service or capability discovery. For example, when a user, User A, scrolls through his/her Address Book and selects a RCS-e contact, the client performs an instant service capability check, being able to display the services which are available to communicate. This mechanism is implemented using SIP OPTIONS.

In order to specify the RCS-e API model and requirements, this document is based on the work already done in the GSMA RCS group using the Rich Communication Suite RCS-e API Detailed Requirements v1.0 [RCS-API] document as base for this one.

[RCS-API] focuses in a UNI/Long tail API model with the intention to put the threshold at the lowest possible level 1) for "anyone" or any application developer to develop a service/application that embeds one or several RCS enablers; 2) allowing to embed RCS enablers in very lightweight environments (such as pure web browser applications).

1.1 Architecture

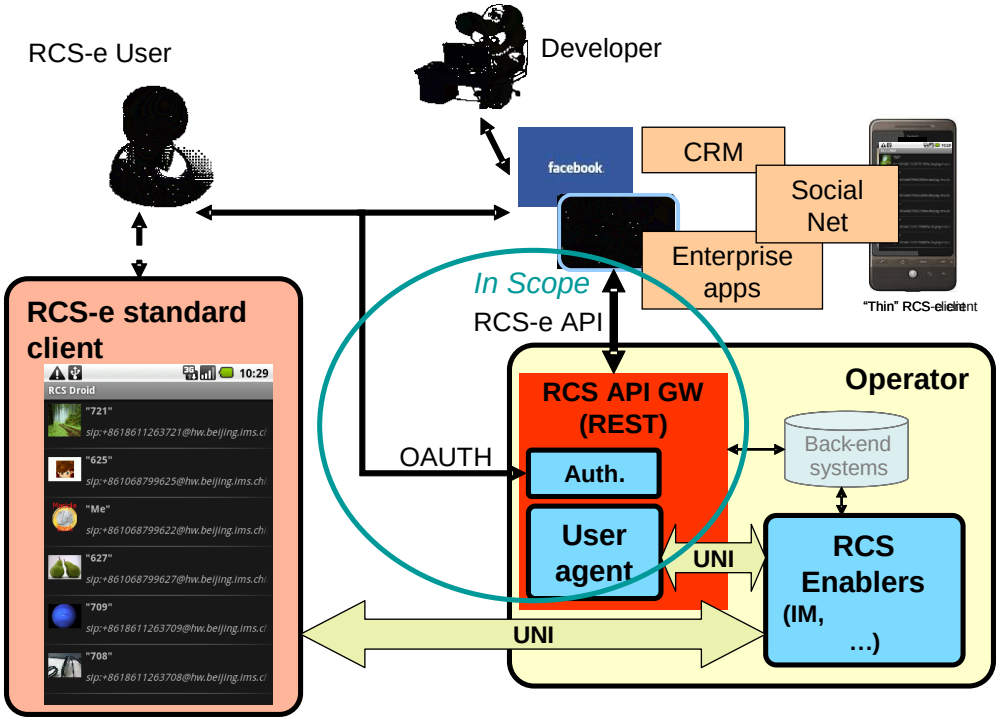

Figure 2 : RCS-e API architecture

Figure 2 shows a sample RCS-e API Architecture supporting:

1. Application authorization to access the RCS-e methods/functions on behalf of the RCS user
2. End-user management of applications user has granted access to, which resource that is granted and the possibility to revoke the access for a given application
3. Operating on the RCS-e user's services via the existing RCS-e UNI using the defined API primitives. (Strictly, as there is no specific RCS-e UNI for voice calls, i.e. call control related APIs do not use RCS-e UNI). If shall be possible also to interact directly with the enablers APIs to provide the same UNI functionality if a performance improvement is achieved.
4. Developer security mechanisms and engagement/registration processes aimed to individual or SME developers (out of scope of this document). Mechanisms and policies shall be defined by the service provider. It is foreseen that in many cases the existing developer portals and communities could accommodate RCS-e.
5. Application and user authentication (out of scope of this document). It is foreseen that in an RCS-e deployment, authentication mechanisms will be defined by the service provider, they could reuse the same authentication used for "regular" clients.

2. API FUNCTIONAL REQUIREMENTS

The functional requirements for the APIs as covered in [RCS-API] document are the following:

1. General requirements
2. Network Address Book UNI API requirements
3. Presence UNI API requirements
4. Messaging (SMS/MMS) UNI API requirements
5. Chat UNI API requirements
6. File Transfer UNI API requirements
7. Call UNI API requirements
8. Video Share and Image Share UNI API requirements

The requirements from RCS not applicable to RCS-e are the ones referring to the Network Address Book, Presence and SMS/MMS APIs.

The new API requirements introduced by RCS-e are the following:

1. Chat UNI API requirements extension to cover message notifications and new session model.
2. Capability Discovery UNI API requirements.

2.1 RCS-e high-level requirements

Label	Description	Comment
UNI-HLF-001	The RCS-e API SHALL be HTTP/REST based.	
UNI-HLF-002	Resource URLs and primitives names SHALL have an intuitive relationship with the functions and resources they are intended to represent.	
UNI-HLF-003	It SHOULD be possible to reuse the Data definitions of the RCS-e APIs for future bindings.	
UNI-HLF-004	The RCS-e APIs SHALL allow including the API version in the resource URLs	
UNI-HLF-004B	The RCS-e APIs SHALL allow an Application and a Server to negotiate the version of a particular resource	This requirement might use the API version in the URL or not.
UNI-HLF-005	The RCS-e API SHALL expose a functional abstraction at the user level rather than at the level of underlying protocols.	
UNI-HLF-006	The RCS-e API SHALL support ”server”-based application clients and ”device”-based application clients. Instantiation examples include applications running on a Web server (where the user interacts with the application via a web browser), or running on a mobile or fixed device as a ”widget” or as a native application.	
UNI-HLF-007	The RCS-e APIs SHALL support application authorization based on OAuth2.0.	Cf. requirement [UNI-OAU-001] Ref: [OAUTH2.0] Users are expected to be authenticated by their service provider, however the authentication mechanisms for the user and application are out of scope of this document, therefore out of scope for RCS-e APIs.
UNI-HLF-008	Subject to the underlying resource capabilities, the RCS-e APIs SHALL NOT expose the real	

	identities of the user and her/his contacts. In particular, mobile telephone numbers (MSISDNs) or identities SHALL NOT be exposed neither for users nor for their contacts. Subject to service provider policies, only trusted applications will be authorized to know that information.	
UNI-HLF-009	The RCS-e APIs SHALL be restricted to the operations and procedures of the RCS-e UNI as defined by RCS-e.	Applications using the RCS-e APIs should not be able to perform operations not possible to a regular RCS client. Ref: [RCS-e spec], Strictly speaking, as there is no specific RCS UNI for voice calls, call control related APIs do not use RCS-e UNI (see section 5.7).

Note: In the context of this section, “widget” should be understood in a loose way as to denote a range of device software ranging from web applets to small non-native applications.

2.2 Authorization framework

Authentication (of user, application, or developer) is out of the scope for this document, because it is foreseen that in an RCS-e deployment authentication mechanisms will be defined by the service provider, typically re-using the authentication used for “regular” RCS-e clients.

Note: In the context of this section, “widget” should be understood in a loose way as to denote a range of device software ranging from web applets to small non-native applications.

2.2.1 *General requirements*

Label	Description	Comment
UNI-AUT-001	The Authorization framework SHALL enable a user owning network resources exposed by a RESTful API to authorize third-party applications to access these resources via this RESTful API on that user’s behalf.	
UNI-AUT-002	The Authorization framework SHALL support network-side Web applications, accessed from user’s Web browser.	
UNI-AUT-003	The Authorization framework SHOULD support client-side stand-alone widget applications installed on user’s terminal, and running outside of a Web browser.	
UNI-AUT-004	The Authorization framework SHOULD support client-side native code applications installed on user’s terminal.	
UNI-AUT-005	The Authorization framework SHALL NOT require a user to reveal to third-party applications the credentials he/she uses to authenticate to the service provider.	Note: This is an RCS-e user privacy requirement.
UNI-AUT-006	The Authorization framework SHALL allow a third-party application to obtain from a service provider (e.g. by provisioning or dynamic discovery) the parameters required to request user’s authorization and to access user’s network resources.	
UNI-AUT-007	The Authorization framework SHALL support a third-party application to initiate the authorization request by directing the user to the service	

	provider's portal.	
UNI-AUT-008	The Authorization framework SHALL support presenting the third-party application's authorization request to the resource owner in a form of an explicit authorization dialog or user consent request.	It is assumed that the user has authenticated to the service provider, before granting authorization (user authentication is out of scope of the Authorization framework). Note: Design and handling of this dialog are out of scope for the RCS-e API. However, the API needs to communicate the parameters needed for the dialog, and/or specified by the user in the dialog
UNI-AUT-009	The Authorization framework SHOULD facilitate presenting to the resource owner at least the third-party application identity, the resources and the operations on these resources for which authorization is requested.	Note: Design and handling of the dialog presenting this are out of scope for the RCS-e API. However, the API needs to communicate the parameters needed for the dialog, and/or specified by the user in the dialog.
UNI-AUT-010	The Authorization framework SHALL enable the resource owner to authorize or deny access to each of the requested resources and operations.	
UNI-AUT-011	The Authorization framework MAY enable the resource owner to specify the duration for which his/her access authorization is granted.	
UNI-AUT-012	The Authorization framework SHOULD facilitate communicating the resource owner's preferred language and terminal capabilities.	
UNI-AUT-013	In case the user authorizes the third-party application to access the user's resources, the Authorization framework SHALL be able to provide to the third-party application an access token representing this user's authorization subject to obtaining it from the issuer.	
UNI-AUT-014	The access token SHALL only be usable by the third-party application for the restricted scope (operations on resources) authorized by the user at the time of authorization request.	
UNI-AUT-015	The Authorization framework SHALL enable the user to invalidate at any time (e.g. upon user's request on service provider's portal) an access token representing a user's authorization to a third-party application.	Note: The operation would be carried out in an out-of-band communication channel (e.g. user web page, policy triggers). EDITOR NOTE: Removed, partially because in conflict with latest OAuth revocation draft (http://datatracker.ietf.org/doc/draft-lodderstedt-oauth-revocation/)
UNI-AUT-016	The Authorization framework SHALL support the inclusion of an access token (e.g. obtained by the third-party application from the service provider for the scope of this request) in requests to resources exposed by the RESTful API.	
UNI-AUT-017	The Authorization framework SHOULD facilitate the possibility to retrieve the list of the third-party applications that have been authorized before and which resources have been authorized per third-party application by the user.	
UNI-AUT-018	The Authorization framework SHOULD facilitate the possibility for the user to remove the authorization for any third-party application that has previously been authorized.	

UNI-AUT-019	Notifications sent to the third-party application SHALL be filtered based on authorization granted to the third-party application, so server SHALL NOT send notifications regarding a resource for which the application has no authorization.	Cf. requirement [UNI-NTF-005]

2.2.2 *Authorization using OAuth*

Label	Description	Comment
UNI-OAU-001	The Authorization framework SHALL be based on OAuth 2.0 as specified in [OAUTH20]	Cf. requirement [UNI-HLF-007] Ref: [OAUTH20]
UNI-OAU-002	The Authorization framework SHALL support the OAuth 2.0 “Authorization Code flow”, where the third-party application is a server-side web application.	
UNI-OAU-003	The Authorization framework SHALL support OAuth 2.0, where the types of third-party applications can either be client-side installed widget applications or client-side native code applications.	
UNI-OAU-004	For the delivery of authorization code (“Authorization Code Flow”) / access token (“Implicit Grant Flow”) to a client-side installed application (widget or native code application), the Authorization framework SHALL support at least one OS-agnostic and application-type agnostic delivery mechanism, which does not require end-user interaction such as manual input of authorization code.	See Annex 1 of [RCS-API] for an informative example of such mechanism, based on binary-SMS. An alternative option would be to use the notification channel as the delivery mechanism.
UNI-OAU-005	The Authorization framework MAY support OAuth 2.0 flows other than the “Authorization Code Flow”.	
UNI-OAU-006	The Authorization framework SHALL support the OAuth 2.0 “Authorization Server” and “Resource Server” roles.	
UNI-OAU-007	The Authorization framework SHALL regard the users resources accessed via the RESTful API as the OAuth 2.0 “Protected Resource”.	
UNI-OAU-008	When following the Authorization Code Flow the Authorization framework SHALL generate a OAuth 2.0 authorization code as a result of the user authorization.	If other flows are used, a similar functionality should be provided.
UNI-OAU-009	The Authorization framework SHALL support the exchange of an authorization code for an access token according to OAuth 2.0.	
UNI-OAU-010	The Authorization framework SHALL bind the authenticated user identity to the generated authorization code and access token.	Note: The actual authentication mechanism used is out of the scope for this document, because it is foreseen that in an RCS-e deployment authentication mechanisms will be defined by the service provider, typically re-using the authentication used for “regular” RCS-e clients.
UNI-OAU-011	The Authorization framework SHALL be able to determine the user identity (for example MSISDN) from the access token received from the application.	Note that in deployments this feature may not be available, or only available to privileged applications, in order to support privacy (e.g.

		using a service provider policy). See also UNI-HLF-008.
UNI-OAU-012	The Authorization framework SHALL validate the access token received from the application according to OAuth 2.0.	
UNI-OAU-013	The values of the OAuth 2.0 “scope” parameter SHALL reflect selected granularity in the usage of RCS enablers/resources via the REST API.	
UNI-OAU-014	The values of OAuth 2.0 “scope” parameter SHALL have a direct mapping (1-to-1 or 1-to-many or many-to-many) to the available RCS-e APIs primitives.	
UNI-OAU-015	The following minimum set of "scope" values targeted granularity SHALL be supported: ・ Chat ・ File transfer ・ Video share ・ Image share ・ Voice call ・ Voice call notification ・ Service capability	API design should assign one of these scope values to each operation defined in the APIs. Note that the mandatory requirement applies only to the targeted granularity of the “scope values” and not necessarily to the listed identifiers themselves. The way the identifiers are specified is left to the technical specification.
UNI-OAU-016	In addition to the values defined in requirement [UNI-AUT-015], it SHOULD be possible to define per-service provider values of “scope” parameter to accommodate different granularity levels.	

Note: all figures are informative.

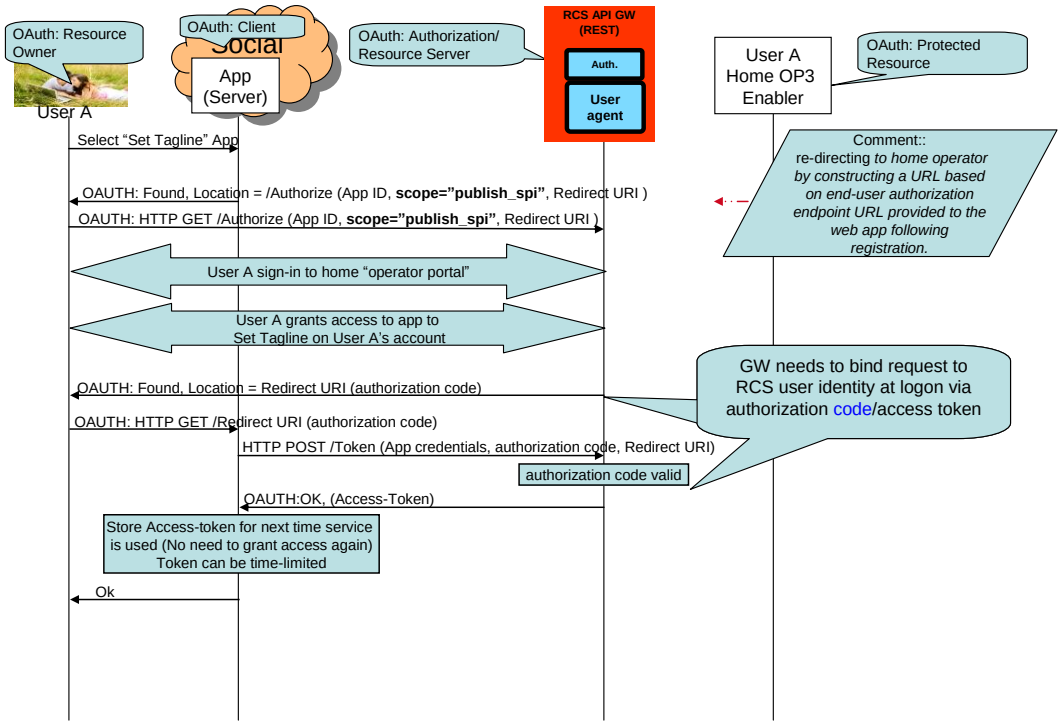

Figure 3: Example of application authorization of OAuth 2.0 in RCS-e using OAuth Authorization Code flow

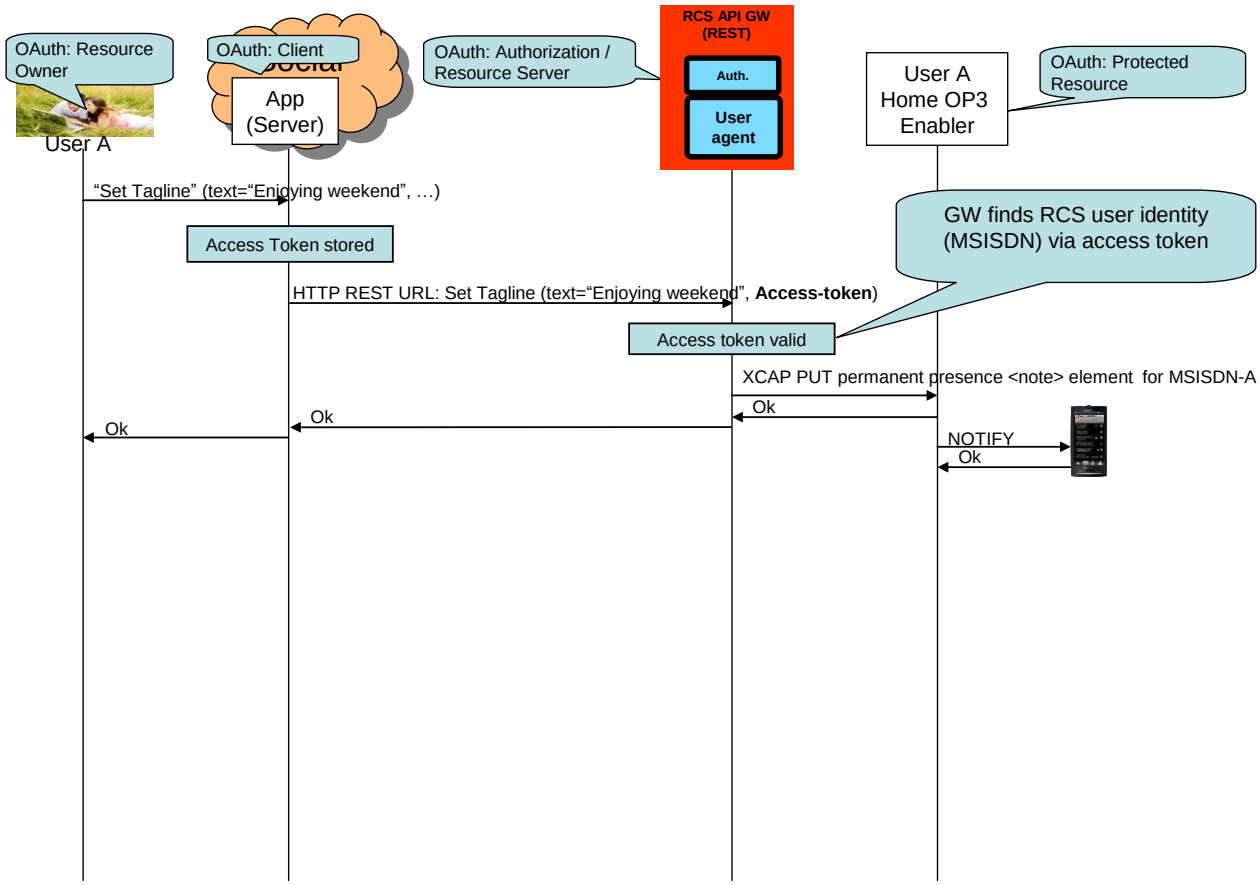

Figure 4 : Example of application usage of OAuth 2.0 in RCS-e

2.3 Common notification cannel

Label	Description	Comment
UNI-NTF-001	The RCS-e APIs SHALL support a common notification mechanism that allows delivery of notifications for multiple different subscriptions to the same endpoint at the application.	Different RCS services needs to alert a user of events (incoming chat invite, presence update from buddy etc.). If each RCS service would have their own notification channel, a multi-service application would need to manage multiple such notification channels. This would result in increased complexity and would be impossible to manage in some environments (as an example, web browsers have a limitation in the number of open HTTP connections). Similar requirements from disparate domains have driven the development of so called bidirectional HTTP technologies (Comet, Reverse AJAX, long polling….), see [RFC6202].
UNI-NTF-002	The RCS-e APIs SHALL support the delivery of notifications directly to an application-defined endpoint, i.e. a callback URL, using HTTP.	The application establishes a subscription to notifications by providing a call-back URL where the notifications are to be received. This method follows the well-known subscription/notification pattern using

		REST primitives. It is foreseen to be used mainly for server-to-server notifications. Emerging industry standards for such notifications like pubsubhubbub (http://code.google.com/p/pubsubhubbub/) could be taken into consideration.
UNI-NTF-003	The RCS-e APIs SHALL support the delivery of notifications to the application in an HTTP-based notification channel using the long-polling mechanism (see [RFC6202]).	This method is foreseen to be used mainly in environments that cannot receive requests from the network or cannot support server environments, such as browsers, devices, set top boxes, and so on. The application issues a “long” polling request to establish a notification channel for receiving notifications.
UNI-NTF-004	The notification mechanisms according to requirement [UNI-NTF-002] and [UNI-NTF-003] SHALL use the same data format and schemes for notifications.	
UNI-NTF-005	Notifications sent SHALL be filtered based on authorization granted to the application, so server SHALL NOT send notifications regarding a resource for which the application has no authorization.	Cf. requirement [UNI-AUT-019]
UNI-NTF-006	The RCS-e APIs SHALL support selective subscriptions of the application to notifications about specific events.	As an example, an application that only reads / sets the free text field is probably not interested on Video Share-related notifications, or contact list update notifications.
UNI-NTF-007	The RCS-e APIs SHALL be able to deliver multiple events in one single (long polling) notification.	This mechanism is foreseen to be used for long-polling but might be adopted in other cases e.g. delivering notifications with a callback URL
UNI-NTF-008	The RCS-e APIs SHALL support the inclusion of a reference to the relevant resource in the notification.	The application can use the received resource reference to perform relevant actions on the resource (e.g. accept invite or get presence data from buddies). Notification events are expected to be able to include details where applicable (e.g. session progress information such as “Chat answered”). EDITOR’s NOTE: It is foreseen that some events will be self-contained, meaning they contain all information the application requires for further processing. Others notifications might require querying a resource, which requires the URL to be included in the event notification.
UNI-NTF-009	RCS-e APIs SHOULD include an informative description or reference model for the “long polling” notification channel.	As there are no telco-related standards using these techniques, to facilitate interworking and guide implementations, including aspects such as when connections should be closed, open or retried. Recommendations and best practices in [RFC6202] for “long polling” to be considered.

2.3.1 *Examples (informative)*

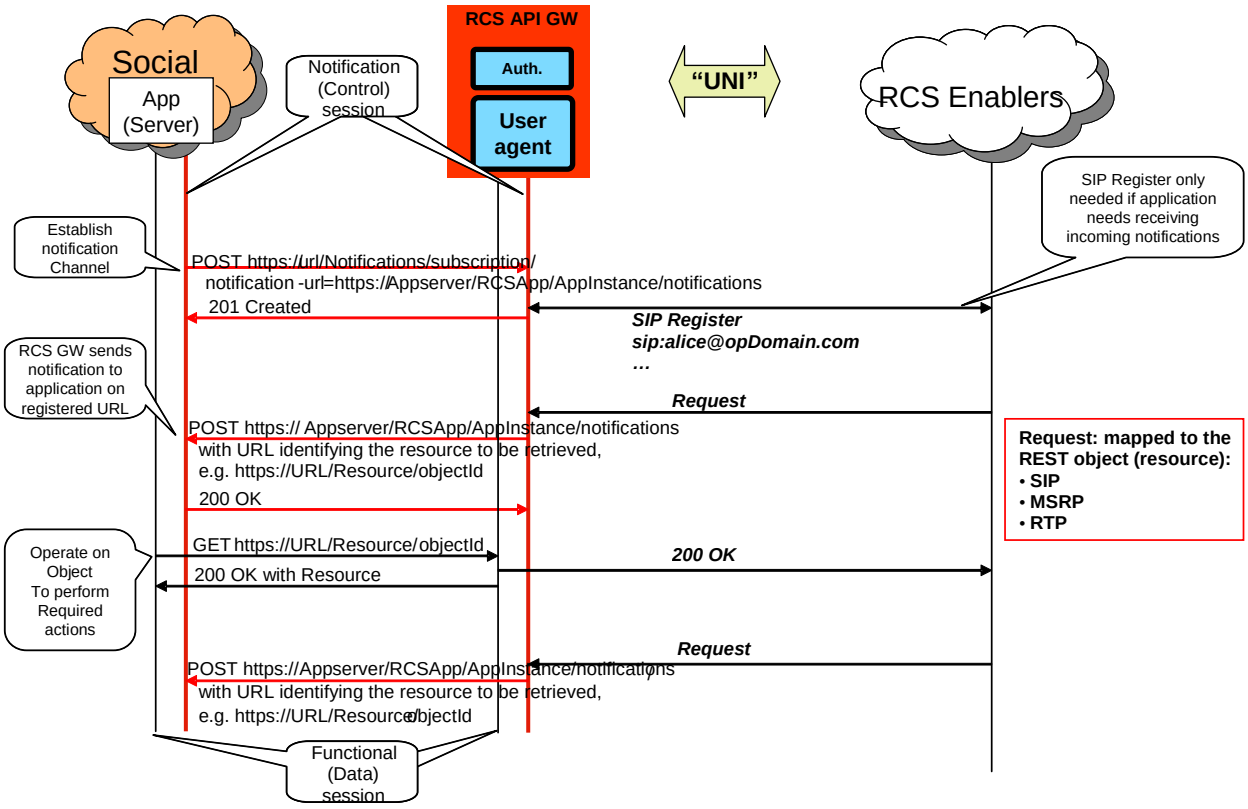

Figure 5 : Notification channel using "subscription" method, example

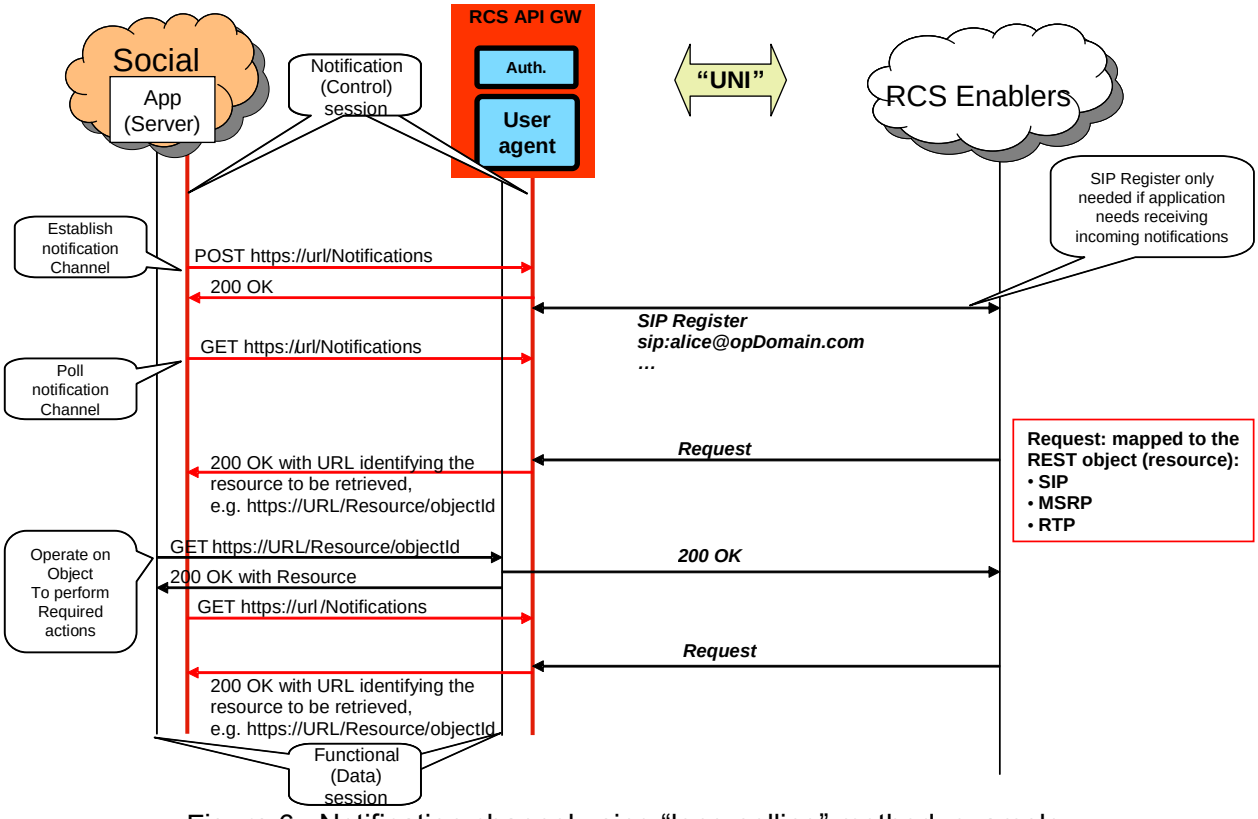

Figure 6 : Notification channel using "long polling" method, example

2.4 Anonymized Customer Reference (ACR) API Requirements

The API gateway providing the RCS-e APIs SHALL NOT expose the real identities of the user and her/his contacts (see UNI-HLF-008). This means that the API will need to use anonymized customer references (ACRs).

Nevertheless, some applications do hold the real identities of their users as they get contact data from other sources (e.g. terminal address books, direct user input, service provider address books). Therefore, a mechanism to translate real identities (e.g. MSISDNs) into ACRs is needed and shall be provided by gateway.

Label	Description	Required parameters	Comment
UNI-ACR-001	The ACR API SHALL support requesting an anonymized customer reference (ACR) associated to an MSISDN.	oauth_token={access-token} msisdn: {msisdn} return value: acr;{anonymized customer reference}	The ACR needs to be stable for a given MSISDN (and application id if applicable), that means that the anonymized id returned by the API shall not change over the time for a given MSISDN and application. For security and end user privacy reasons, it is recommended that the ACRs for a given MSISDN varies with the application id. That is, it is recommended that two different applications get different anomymized ids for the same MSISDN. For MSISDN, the tel: URI scheme [RFC3966] SHOULD be used in the interface for an MSISDN; and the acr: URI scheme ([ACRDRAFT]) SHOULD be used for the anonymized customer reference.

2.5 Contact data and Network Address Book

This chapter has an informative character. It captures the discussion of the working group on contact data and network address books.

Contact data is essential for RCS-e communication. An RCS-e application can get contact data from different sources:

1. Direct user input
2. Terminal address book
3. Network address book of an RCS-e API provider
4. Network address book of a service provider that does not offer the RCS-e API

The interfaces via which the address book is accessed by the application are implementation-specific.
However, MSISDN or an anonymized identifier is needed to link to an RCS user.

For RCS-e API providers that also run a network address book, it is recommended that the address book works with the anonymized customer references as specified in this document.

2.6 Capability Discovery API Requirements

2.6.1 *Capability Discovery*

As part of the new RCS-e specification the capability discovery is one of the key functionalities and shall be exposed by the RCS-e API gateway.

Apart of the standard RCS-e capabilities applications created using the APIs shall be able to register and exchange new capabilities in order to be able to now when other user supports that application.

The following table describes the UNI API requirements for the capability discovery:

Label	Description	Required parameters	Comment
UNI-CPD-001	The Capability Discovery API shall be able to register a new service capability feature tag related to the application. This capability shall be enabled by UNI-CPD-003 before being exposed by the application on behalf the user.	oauth_token={access-token} capability: {capability_id}	Use case: Game application using RCS-e to discover which contacts are also available for gamming. Note: Registering new application feature tags is subject to operator policies.
UNI-CPD-002	The Capability Discovery API shall be able to unregister a previously registered capability feature tag related to the application.	oauth_token={access-token} capability: {capability_id}	
UNI-CPD-003	The Capability Discovery API shall be able to enable or disable any standard RCS-e capability or a custom application registered capability.	oauth_token={access-token} enabled:{true/false} capability: {capability_id}	The network element providing this APIs should answer any user capability user request (via OPTIONS) returning only the feature tags related to the enabled capabilities.
UNI-CPD-004	The Capability Discovery API shall allow an application to query the service capabilities of a certain contact.	oauth_token={access-token} contact:{contact_id}	

2.6.2 *User Discovery*

User discovery supports an application to find out which of an user's contacts are RCS-e enabled. This API is typically called when an application initializes its address book.

Label	Description	Required parameters	Comment
UNI-CPD-004	The Capability Discovery API shall allow an application to query if a certain contact is RCS-e capable or not.	oauth_token={access-token} contact: {contact_id} Return value: {true, false}	

2.7 Chat UNI API requirements

2.7.1 *One to One Chat*

According to the RCS-e specification:

There is no explicit chat invitation associated to the one to one chat anymore. From the functional point of view the user sends a message to another user and it is responsibility of the client implementation to open any underlying SIP/MSRP session to deliver that message. This complexity is hidden to the user.

Also from the receiver point of view, the user does not accept or decline a one to one chat invitation, he just receives a new message from a user. So, there is no way a user is able to accept or reject an SIP/MSRP session from the client application, and the establishment mechanism is controlled by the client application according to the MNO rules, as seen in RCS-e technical specification chapter 3.2.4.2 “Answering a chat”.

Due to that fact, no functional requirements associated to one to one chat establishment (for either the originating or terminating side) are required for RCS-e.

Also information regarding the technical establishment or ending of the underlying IM session (i.e. SIP and MSRP session) are out of scope of this API specification.

The only requirements applicable then to the 1 to 1 chat are the ones related to the media and the notifications.

All the requirements present in the RCS API document related to originating and terminating side, chapters 4.5.1 and 4.5.2 in [RCS API], does not apply anymore for one to one chat and therefore have been moved to the group chat chapter.

2.7.2 *Group chat*

The operations below allow managing a group chat.

Label	Description	Required parameters	Comment
UNI-CHT-001	VOID		Not applicable for RCS-e. Numbering keep for editorial consistency.
UNI-CHT-002	The Chat API SHALL support starting a group chat, where the initial chat message SHALL be included in the subject field	oauth_token={access-token} recipient={contact1}, {contact2}, … subject={text} (e.g. “Hi”)	Use case: Start a group chat (ad-hoc group). Conference focus id must be returned to application Ref: [RCSR2TR] ch 10.2.1.1 , [RCSR3IMEND] ch 7.1.1.3
UNI-CHT-003	The Chat API SHALL support cancelling a group chat invitation	oauth_token={access-token}	Use case: user cancels a chat invitation. Cancellation is only possible as long as the invitation has not been accepted. Ref: [RCSR3IMEND] ch 7.1.1.13
UNI-CHT-004	The Chat API SHALL support notifications about group chat (accepted, cancelled; declined, ended)	oauth_token={access-token}	
UNI-CHT-005	VOID		Not applicable for RCS-e. Numbering keep for editorial consistency.
UNI-CHT-006	VOID		Not applicable for RCS-e. Numbering keep for editorial consistency.
UNI-CHT-007	The Chat API SHALL support notifications about incoming chat invite	Information about inviting user, subject header, and other invited participants (in case of group chat).	Use case: user is invited to a chat session. It might be possible that the inviting user is not in the contact list. See “Common notification channel”

			for establishment of notification channel. Ref: [RCSR2TR] ch 10.2.1.2, [RCSR3IMEND] ch 7.1.2.1
UNI-CHT-008	The Chat API SHALL support accepting a group chat invitation	oauth_token={access-token}	Use Case: User accepts a group chat invitation. Ref: [RCSR2TR] ch 10.2.1.2, [RCSR3IMEND] ch 7.1.2.1 NOTE: Only applicable to group chat invitations in RCS-e.
UNI-CHT-009	The Chat API SHALL support declining a group chat invitation	oauth_token={access-token}	Use Case: User declines a group chat invitation. Ref: [RCSR2TR] ch 10.2.1.2, [RCSR3IMEND] ch 7.1.2.1 NOTE: Only applicable to group chat invitations in RCS-e.
UNI-CHT-010	VOID		Not applicable for RCS-e. Numbering keep for editorial consistency.
UNI-CHT-011	The Chat API SHALL support leaving a group chat	oauth_token={access-token}	Use Case 1: User leaves a group chat. This ends the chat for this user. Use Case 2: If group chat originating user leaves the group chat, depending on the operator policies the group chat session could be terminated or not. Ref: [RCSR3IMEND] ch 7.1.1.12
UNI-CHT-012	The Chat API SHALL support notifications about “group chat ended”		In case of group chat termination the users will receive a notification about that event. See use case 2 of previous requirement. See “Common notification channel” for establishment of notification channel. Ref: [RCSR3IMEND] ch 7.1.2,3
UNI-CHT-013	VOID		Not applicable to RCS-e.
UNI-CHT-014	The Chat API SHALL support adding a set of users to a group chat	oauth_token={access-token} recipient={contact1}, {contact2}, …	Use Case: User adds one or more participants to the group chat. The new participant(s) receive(s) a chat invitation. Ref: [RCSR3IMEND] ch 7.1.1.7
UNI-CHT-015	The Chat API SHALL support re-joining a group chat	oauth_token={access-token} chat conference id={sessionid}	Use Case: User wants to join a chat (possible use cases: invitation has expired, user left and wants to rejoin, and so on). As a result, user successfully re-joined chat (if chat/session found), or alternatively an indication is returned that chat/session not found (due to expiry). Ref: [RCSR3IMEND] ch 7.1.1.9
UNI-CHT-016	The Chat API SHALL support subscribing to notifications about participant information in a group chat	oauth_token={access-token}	See “Common notification channel” for establishment of notification channel. Ref: [RCSR3IMEND] ch 7.1.1.11
UNI-CHT-017	The Chat API SHALL support notifications about participant information in a		Use case: The application receives notifications about the changing set of participants in a group chat

	group chat. A notification SHALL be generated upon subscription, as well as when the set of participants changes		session. See “Common notification channel” for establishment of notification channel. Ref: [RCSR3IMEND] ch 7.1.1.11

2.7.3 *Media*

The operations below allow handle the media in a chat.

Label	Description	Required parameters	Comment
UNI-CHT-018	The Chat API SHALL support sending messages and acknowledging the successful sending of those messages	oauth_token={access-token} message_content={content} return: status: {success,pending}	Use case: The application sends a chat message. Content can be only text according to RCS-e specifications. If the message was successfully delivered to the next hop a “success” response is returned. In case the transaction is taking too much time to be completed it shall be possible to return a “pending” response and return the final delivery status asynchronous via the notification channel. Ref: [RCSR3IMEND] ch 7.1.3.2
UNI-CHT-019	The Chat API SHALL support send “isComposing”	oauth_token={access-token} isComposing=“active”/”idle” “timeout=xx”” …	Use case: The application sends “isComposing” which indicates that a user is currently composing a message. Same as [UNI-CHT-018] with “isComposing” as a special kind of content, parameters according to RFC 3994. If the message delivery was successful a “success” response is returned. Ref: [RCSR3IMEND] ch 7.1.3.4
UNI-CHT-020	The Chat API SHALL support receiving messages	oauth_token={access-token}	Use case: the application receives a chat message via the notification mechanism. Timestamp value SHALL be also notified to the application if it was included in the message. [Add: Store & Forward use case] See “Common notification channel” for establishment of notification channel. Ref: [RCSR3IMEND] ch 7.1.3.3
UNI-CHT-021	The Chat API SHALL support receiving the “isComposing” message	oauth_token={access-token}	Use case: the application receives via the notification mechanism an indication that a user is currently composing a message. Same as [UNI-CHT-020] with “isComposing” as a special kind of content. Ref: [RCSR3IMEND] ch 7.1.3.5

2.7.4 *Notifications*

In RCS-e specification, three notifications associated to messages have been specified:

- "Sent" notification, which is generated when the RCS-e client has successfully send the message to the next hop (i.e. IM Server if store and forward is enabled on the network). In the case of the APIs it should be generated by the API gateway and notified to the application when it successfully has sent the message.

- "Delivery" notification, which is generated when the message arrives to the final destination. In the case of the APIs, the API gateway will receive the notification from the IM Server about a previously sent message and it will notify the application accordingly. The API gateway is also responsible of sending back the delivery notifications of incoming messages as they are received by the application. In order to avoid sending delivery notifications for messages that are not correctly received (i.e. the application fails to fetch the message while it is in the notification channel), it is highly recommended that the API gateway sends the "delivery" notification for incoming messages only after the message has been successfully delivered to the application in the notification channel.

- "Displayed" notification, which is generated by the RCS-e client when a message is displayed on the RCS-e device. For privacy issues, an RCS-e user is able to enable or disable the sending of "displayed" notifications. In the case of APIs, the application is the responsible of generating these "displayed" notifications accordingly and also the API gateway shall be able to receive them and notify the application.

For group chat the message notifications are not required.

The operations below allow handle the message related notifications.

Label	Description	Required parameters	Comment
UNI-CHT-022	The Chat API SHALL support receiving messages notifications ["sent", "delivered" and "displayed"] for messages sent in a 1 to 1 session.		Message notifications SHALL be returned asynchronously via the notification channel except the "sent" notification. As stated in the UNI-CHT-018 the "success" response is returned it SHALL be considered as the sent notification.
UNI-CHT-023	The Chat API SHALL support sending "displayed" notifications of 1 to 1 message received.	oauth_token={access-token} message id={message-id}	The message-id shall be the one received in the incoming message.

2.8 File Transfer UNI API requirements

The following tables show the functional requirements for the file transfer API.

2.8.1 *Originating side*

Label	Description	Required parameters	Comment
UNI-FLT-001	The File Transfer API SHALL support initiating a file transfer	oauth_token={access-token} recipient={contactid} file-icon={reduced image} file-name={file name} file-size={size} file-type={type} url={url to the file}	Initiate a file transfer session with the selected recipient. A SIP INVITE request is sent to the remote party (the contact). A file transfer instance is created at reception of indication that invite & initial message were delivered (SIP 180).

		or BODY{image file}	The file could be send either in the body of the request or send an url to the actual file. Ref: R3 FD 3.4.2, [RCSR3IMEND] ch 10.1
UNI-FLT-002	The File Transfer API SHALL support cancelling a file transfer invitation by the originating side	oauth_token={access-token}	Use case: An ongoing file transfer session is to be cancelled. Only the user that created the invitation can cancel it, and it is only offered before the file transfer is accepted or rejected. Ref: [RCSR3IMEND] ch 10.1
UNI-FLT-003	The File Transfer API SHALL support ending a file transfer session by the originating side	oauth_token={access-token}	The selected resource, that is, the file transfer session, is to be closed. A SIP BYE request for the selected session is sent to the remote party. Ref: [RCSR3IMEND] ch 10.2
UNI-FLT-004	The File Transfer API SHALL support notifications about “File Transfer” (accepted, declined, cancelled, ended) to the originating side		The final set of applicable notification types will be determined in the technical work phase. See “Common notification channel” for establishment of notification channel.

2.8.2 *Terminating side*

Label	Description	Required parameters	Comment
UNI-FLT-005	The File Transfer API SHALL support notifications about file transfer invitation		Use case: user is invited to a file transfer session. See “Common notification channel” for establishment of notification channel. Ref: [RCSR3IMEND] ch 10.3
UNI-FLT-006	The File Transfer API SHALL support accepting a file transfer invitation by the terminating side	oauth_token={access-token}	Use case: File transfer session is to be accepted. Ref: [RCSR3IMEND] ch 10.3
UNI-FLT-007	The File Transfer API SHALL support declining a file transfer invitation by the terminating side	oauth_token={access-token}	Use case: File transfer session is to be rejected. The SIP INVITE request is then rejected with a SIP 603 response. Ref: [RCSR3IMEND] ch 10.3
UNI-FLT-008	The File Transfer API SHALL support ending a file transfer by the terminating side	oauth_token={access-token}	Use case: File transfer session is to be closed. A SIP BYE request for the selected session is sent to the remote party. Ongoing file transfer can only be cancelled once the session is established. Ref: [RCSR3IMEND] ch 10.1
UNI-FLT-009	The File Transfer API SHALL support final state notifications about the MSRP transfer session (“success”, “abort” and “error”)to the terminating side		The final set of applicable notification types will be determined in the technical work phase. See “Common notification channel” for establishment of notification channel.
UNI- FLT-010	The File Transfer API SHALL support notifications indicating that the file transfer content is available for download	url={file url}	The gateway will send this notification to the client with url to download the image. The url SHALL be ready to start downloading when the notification is

			sent. It is up to the implementation to decide if this is sent when the first chunks of MSRP data are received and allow to simultaneously receiving data from the MSRP session and HTTP downloading or if it waits for the MSRP session to be completed and only allow the download to be started when the whole file has been received. In any case the notification SHALL be sent before the final state notification is sent.

2.9 Call UNI API requirements

The Call UNI API requirements are based on OMA ParlayREST Third-Party Call Control and Call Notification APIs.

2.9.1 *Call Functionality available to originating side*

The operations below allow an application to manage a call session and to receive call progress notifications on behalf of the originating side (“calling participant”, “A-Party”).

Label	Description	Required parameters	Comment
UNI-CLL-001	The Call API(s) SHALL support initiating a call session with a called party	oauth_token={access-token} recipient={contactid}	Use case: User initiates a call between own terminal and another user. Initiating user’s terminals all ring. User answers on one of his terminals. After this the call is set up to the intended recipient.
UNI-CLL-002	The Call API(s) SHALL support the cancellation of the call session initiation.	oauth_token={access-token}	Use case: User interrupts call attempt.

2.9.2 *Call functionality available to originating side and terminating side*

The operations below allow an application to receive call progress notifications and to terminate a call session on behalf of the call participants (“calling participant”, “A-Party” as well as “called participant”, “B-Party”). The term “user” below therefore subsumes both “A-party” as well as “B-party”.

Label	Description	Required parameters	Comment
UNI-CLL-003	The Call API(s) SHALL support notifications about “call alerting”.		Use case: Application receives call invitation notification that the user’s phone is ringing. See “Common notification channel” for establishment of notification channel.
UNI-CLL-004	The Call API(s) SHALL support notifications about “call accepted”.		Use case: Application receives notification that the user’s phone accepts call. See “Common notification channel” for establishment of notification channel.
UNI-CLL-005	The Call API(s) SHALL support notifications about “busy”.		Use case: Application receives notification that the user’s phone is busy.

			See “Common notification channel” for establishment of notification channel.
UNI-CLL-006	The Call API(s) SHALL support notifications about “not reachable”.		Use case: Application receives notification that the user’s phone is disconnected. See “Common notification channel” for establishment of notification channel.
UNI-CLL-007	The Call API(s) SHALL support notifications about “no answer”.		Use case: Application receives notification that the user’s phone did not react to the call. See “Common notification channel” for establishment of notification channel.
UNI-CLL-008	The Call API(s) SHALL support notifications about “disconnected”.		Use case: Application receives notification that the user’s phone has ended the call. See “Common notification channel” for establishment of notification channel.
UNI-CLL-009	The Call API(s) SHALL support terminating a call session.	oauth_token={access-token}	Use case: The call session is terminated by the application, rather than by one of the call participants on-hooking the phone.
UNI-CLL-010	The Call API(s) MAY support notifications about “call declined”.		Note that this event may or may not be generated by the actual API gateway, depending on the underlying network infrastructure. In a SIP environment, this maps to 603 Decline. See “Common notification channel” for establishment of notification channel.

2.9.3 *Media*

Out of scope.

2.10 Video Share UNI API requirements

References for Video Share: GSMA IR.74 [IR74] as endorsed by RCS-e.

2.10.1 *Video Share use cases (informative)*

To clarify the requirements in the next sections, the intended basic use cases of the Video Share API are:

1. API Originated: Sharing a recorded or stored video file from application to client.

The application acts as an originating client in a Video Share session. For instance, a music television station offers their customers to browse a catalogue of music videos, and stream them by click to clients. The application uses a video file as the source of the video stream of the VS.

For option 1, the file is included as the body of the API request to create the video share session. This ensures that the video file is available when the video share session is accepted.

The method to upload the media file to the repository in option 2 is out of the scope.

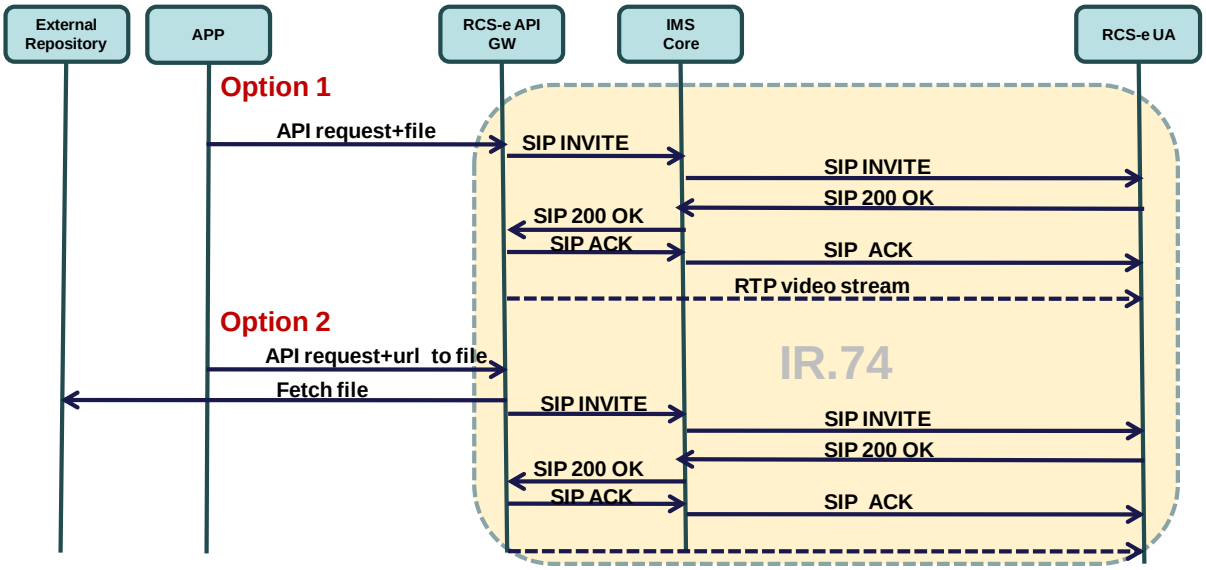

Figure 7: Schematic flow for Video Share use case 1

2. API Originated: Sharing real time video from application to client.

The application acts as an originating client in a Video Share session. For instance, application streams video from a live video feed to clients.

The application creates a new Video Share session and announces to the gateway which formats (transport protocol, codecs, etc) it supports. The gateway processes the list and selects one of the offered formats (transport protocol, codecs, etc). The gateway then makes a Video Share invitation to the IR-74 compliant client. When the client accepts the Video Share session, the gateway sends a notification to the application using the notification channel indicating the chosen format and the media url and/or access parameters, to which the application shall subsequently send the media.

The API will provide an open and extensible mechanism to signal the media formats (transport protocol, codecs, etc), but the specification of the media protocols and connection/play mechanisms are out of the scope of this API specification (marked in green in Figure 8).

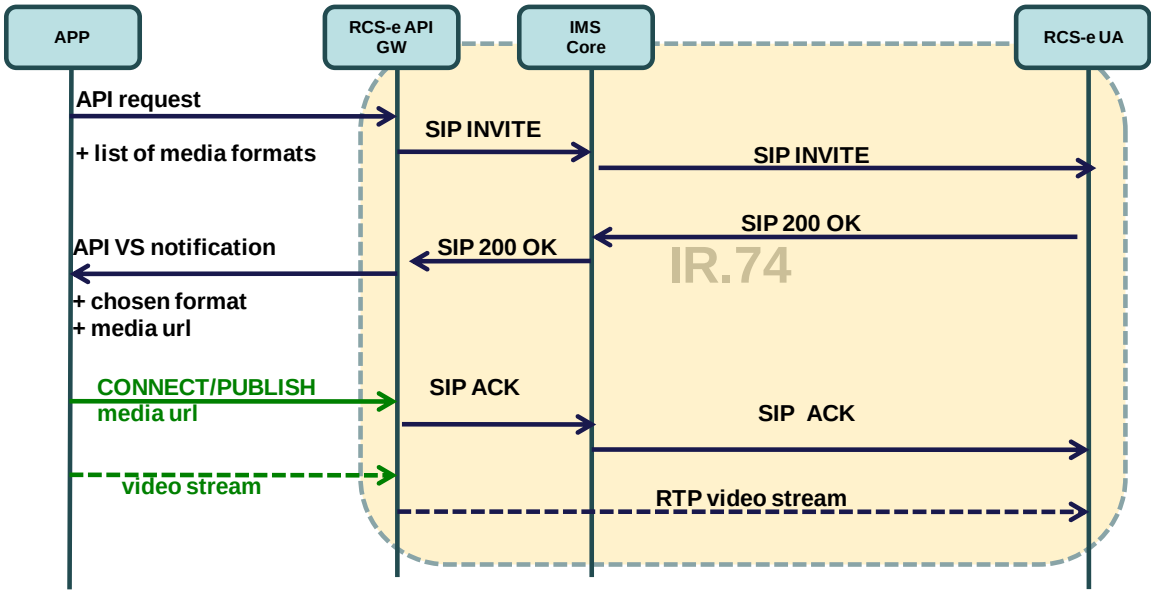

Figure 8: Schematic flow for Video Share use case 2

3. API terminated: Sharing video from client to application

The application acts as a terminating client in a Video Share session. For instance, it could allow a user to watch in real time from a web browser the video that was shared. Another example would be an application that records the shared video for later use.

A summarized interaction would be as follows: The VS is started by an IR.74 compliant handset. The gateway receives the IR.74 invitation, and notifies the application about it indicating a list of formats (transport protocol, codecs, etc...) in which the media can be made available.

The application searches the list for the most suitable format according to the platform/software it is running, accepts the VS session indicating the chosen format. In the response to this acceptance request, the gateway will return the url and/or any other access parameters which the client needs to access the media.

The API will provide an open and extensible signaling mechanism for codecs, formats, transports, etc, but the specification of the media protocols and connection/play mechanisms are out of the scope of this API specification (marked in green in Figure 9).

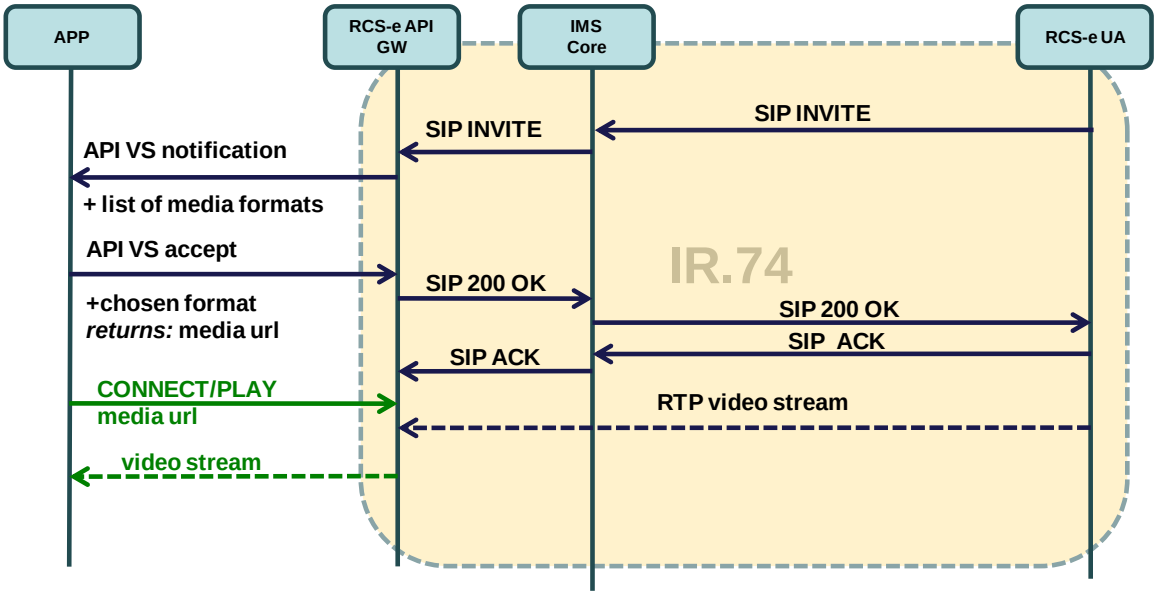

Figure 9: Schematic flow for Video Share use case 3

More complicated use cases can be built composing on these three basic ones. Also note that IR.74 compliant clients can support these three use cases with no changes.

2.10.2 *Video Share functionalities available to originating side*

Label	Description	Required parameters	Comment
UNI-VSH-001	The Video Share API SHALL support initiating a Video Share session using a video file.	oauth_token={access-token} recipient={contactid} or call={callObjectID} url={url to the media file} or BODY{media file}	See use case 1 for more details about this requirement. Arguments need to contain at least either a reference to an existing call or a recipient. When the Video Share is established with the call id, the gateway will link the “initiate Video Share” request to the ongoing call. Video Share object instance is created and returned immediately to accommodate cancelling before alerting. The video file could be send either in the body of the request (option 1) or send an url to the media file (option 2)
UNI-VSH-01b	The Video Share API SHALL support initiating a Video Share session using real time video feed.	oauth_token={access-token} recipient={contactid} or call={callObjectID} formats={list of media formats}	See use case 2 for more details about this requirement. Arguments need to contain at least either a reference to an existing call or a recipient. When the Video Share is established with the call id, the gateway will link the “initiate Video Share” request to the ongoing call.

			Video Share object instance is created and returned immediately to accommodate cancelling before alerting. The application shall send the list of formats (transport protocol, codecs, etc) that it supports.
UNI-VSH-002	VOID	VOID	VOID
UNI-VSH-003	VOID	VOID	VOID
UNI-VSH-004	The Video Share API SHALL support cancelling a Video Share session invitation by the originating side.	oauth_token={access-token}	Use case: Application on originating side interrupts Video Share attempt. Only the user that created the invitation can cancel it.
UNI-VSH-005	The Video Share API SHALL support sending notifications to the application about Video Share state ("accepted", "ended", "declined", "failed")	If "accepted" the notification can include the following information: Chosen media format Media Url	The final set of applicable notification types will be determined in the technical work phase. See "Common notification channel" for establishment of notification channel. In the case the video share session was initiated using a live video feed as indicated in the UNI-VSH-002 requirement, the APIs shall include the chosen format and media url to which the application shall send the media in the "accepted" notification. See use case 2 for more details.
UNI-VSH-006	The Video Share API SHALL support ending Video Share by the originating side.	oauth_token={access-token}	Use case: Application on originating side stops Video Share. A SIP BYE is sent to the remote end.

2.10.3 *Video Share functionality available to terminating side*

Label	Description	Required parameters	Comment
UNI-VSH-007	The Video Share API SHALL support receiving a Video Share invitation	Inviting contact or Reference to an ongoing call List of media formats	See use case 3 for more details on this requirement. The gateway receives the Video Share session invitation, and notifies the application about it indicating a list of formats (transport protocol, codecs, etc...) in which the media can be made available.
UNI-VSH-008	VOID	VOID	VOID
UNI-VSH-009	The Video Share API SHALL support accepting a Video Share session by the terminating side.	oauth_token={access-token} format={format} returns: media_url={media_url} parameters={param1,..}	When user accepts the Video Share session invitation, the application will search the list for the most suitable format according to the platform/software it is running and indicate the chosen format in the acceptance request. In the response to this acceptance request, the gateway will return the url and/or any other access parameters which the client needs to

			access the media.
UNI-VSH-009b	The Video Share API SHALL support rejecting a Video Share by the terminating side.	oauth_token={access-token}	
UNI-VSH-010	The Video Share API SHALL support ending a Video Share by the terminating side.	oauth_token={access-token}	Use case: Application on terminating side ends Video Share. Triggers sending BYE to originating side.
UNI-VSH-011	The Video Share API SHALL support notifications about “Video Share” (“ended”, “cancelled”, “failed”) to the terminating side.		The final set of applicable notification types will be determined in the technical work phase. See “Common notification channel” for establishment of notification channel.

2.11 Image Share UNI API requirements

References for Video Share: GSMA IR.79 [IR79] as endorsed by RCS-e.

2.11.1 *Image Share use cases (informative)*

To clarify the requirements in the next sections, the intended basic use cases of the Image Share API are:

1. API Originated: Sharing a file from application to client.

The IS is started by the application using the API. The application uses an image file as the source of the IS transfer. The image file can be either included in the initial API call or retrieved from an external repository. Method to upload the image file to the repository is outside of the scope.

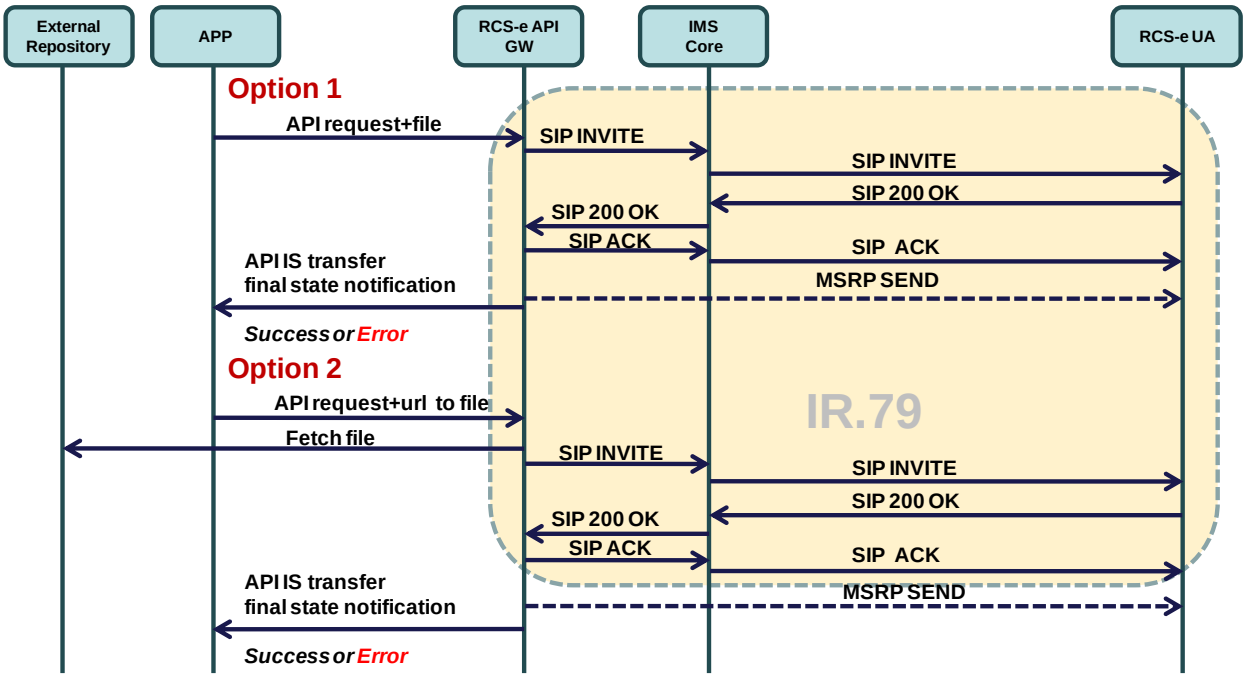

Figure 10: Schematic flow for Image Share use case 1

2. API Terminated: Sharing a file from application to client.
The IS is started by an IR.79 compliant client. The gateway receives the IR.79 invitation, and notifies the application. If the application accepts the invitation, the IS will be established between the GW and the UA. When the IS session is correctly established the application will be notified and given a URL in which the file can be downloaded.

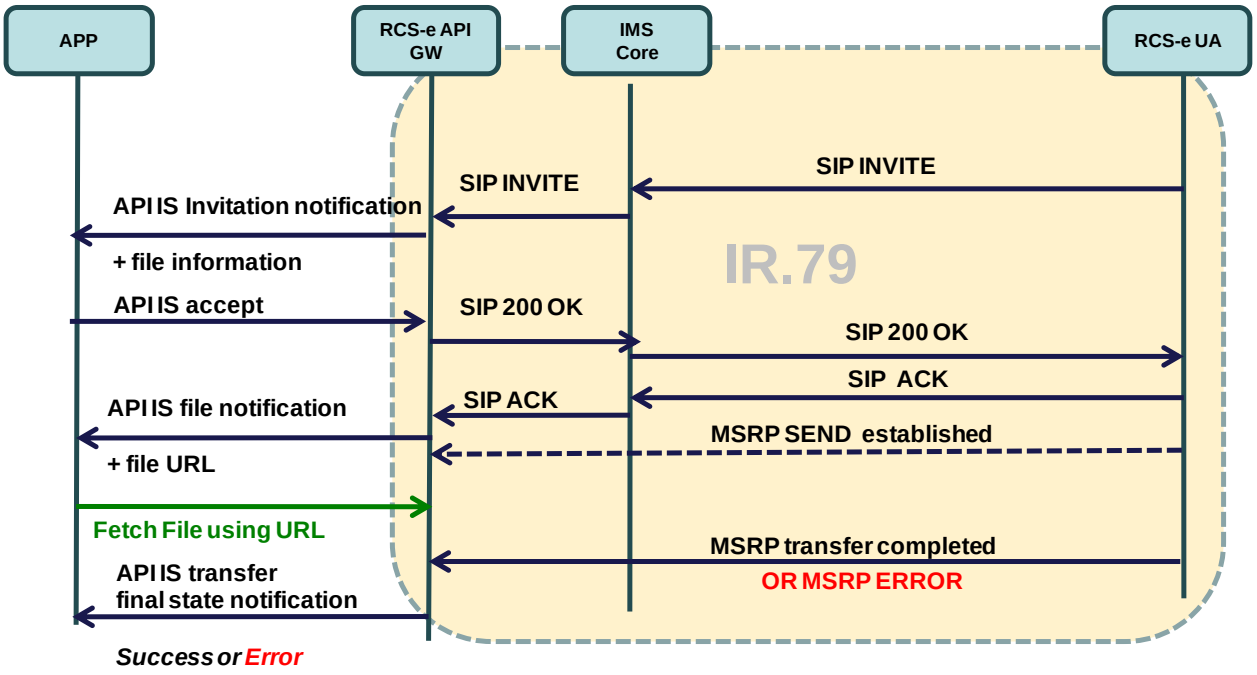

Figure 11: Schematic flow for Image Share use case 2

2.11.2 *Image Share functionality available to originating side*

Label	Description	Required parameters	Comment
UNI-ISH-001	The Image Share API SHALL support initiating a Image Share to a user	oauth_token={access-token} recipient={contactid} call={callObjectID} url={url to the image file} or BODY{image file}	Use case: Application on originating side initiates Image Share. Arguments need to contain at least either a reference to an existing call (callObjectId) for [IR79] Image Share or a Recipient for Image Share without call (i.e. using OMA IM File Transfer). The image file could be send either in the body of the request (option 1) or send an url to the image file (option 2)
UNI-ISH-002	VOID	VOID	VOID
UNI-ISH-003	VOID	VOID	VOID
UNI-ISH-004	The Image Share API SHALL support cancelling an Image Share by the originating side.	oauth_token={access-token}	Use case: Application on originating side interrupts Image Share attempt. It is only offered before the session is accepted.
UNI-ISH-005	The Image Share API SHALL support notifications about Image Share (“alerting”,		The final set of applicable notification types will be determined in the technical work phase.

	"accepted", "ended", "declined", "failed")		See "Common notification channel" for establishment of notification channel.
UNI-ISH-006	The Image Share API SHALL support ending Image Share by the originating side.	oauth_token={access-token}	Use case: Application on originating side stops Image Share. A SIP BYE is sent to the remote end.

2.11.3 *Image Share functionality available to terminating side*

Label	Description	Required parameters	Comment
UNI-ISH-007	The Image Share API SHALL support receiving a Image Share invitation	Inviting contact or Reference to an ongoing call (for IR.79)	Use case: Application on terminating side receives Image Share invitation. See "Common notification channel" for establishment of notification channel.
UNI-ISH-008	VOID	VOID	VOID
UNI-ISH-009	The Image Share API SHALL support accepting or rejecting a Image Share by the terminating side.	oauth_token={access-token}	Use case: Application on terminating side accepts Image Share. Triggers sending a SIP 200 (if accepted) or a suitable rejection cause (if declined) to originating side.
UNI-ISH-010	The Image Share API SHALL support ending an Image Share by the terminating side.	oauth_token={access-token}	Use case: Application on terminating side ends Image Share. Triggers sending BYE to originating side.
UNI-ISH-011	The Image Share API SHALL support final state notifications about the Image Share MSRP transfer session ("success", "abort" and "error").		The final set of applicable notification types will be determined in the technical work phase. See "Common notification channel" for establishment of notification channel.
UNI-ISH-012	The Image Share API SHALL support notifications indicating that the image share content is available for download	url={img url}	The gateway will send this notification to the client with url to download the image. The url SHALL be ready to start downloading when the notification is sent. It is up to the implementation to decide if this is sent when the first chunks of MSRP data are received and allow to simultaneously receiving data from the MSRP session and HTTP downloading or if it waits for the MSRP session to be completed and only allow the download to be started when the whole file has been received. In any case the notification SHALL be sent before the final state notification is sent.

3. REFERENCES

Name	Referenced document
[RCS-e spec]	RCS-e - Advanced Communications: Services and Client Specification Version 1.1-final
[RCS-API]	Rich Communication Suite RCS API Detailed Requirements 1.0 9 March 2011
[RFC6202]	Known issues and best practices for the Use of Long Polling and Streaming in Bidirectional HTTP http://tools.ietf.org/html/rfc6202
[OAUTH20]	The OAuth 2.0 Protocol Framework – Last DRAFT http://tools.ietf.org/html/draft-ietf-oauth-v2
[RCSR1FD]	RCS Release 1 Functional Description http://www.gsmworld.com/our-work/mobile_lifestyle/rcs/rcs_specification_documents.htm
[RCSR1TR]	RCS Release 1 Technical Realization http://www.gsmworld.com/our-work/mobile_lifestyle/rcs/rcs_specification_documents.htm
[RCSR3FD]	RCS Release 3 Functional Description http://www.gsmworld.com/our-work/mobile_lifestyle/rcs/rcs_specification_documents.htm
[RCSR3TR]	RCS Release 3 Technical Realization http://www.gsmworld.com/our-work/mobile_lifestyle/rcs/rcs_specification_documents.htm
[RCSR2TR]	RCS Release 2 Technical Realization http://www.gsmworld.com/our-work/mobile_lifestyle/rcs/rcs_specification_documents.htm
[RCSR3IMEND]	RCS Release 3 OMA IM Endorsement http://www.gsmworld.com/our-work/mobile_lifestyle/rcs/rcs_specification_documents.htm
[IR74]	GSMA IR.74 - Video Share Interoperability Specification http://gsmworld.com/documents
[IR79]	GSMA IR.79 - Image Share Interoperability Specification http://gsmworld.com/documents
[IR84]	GSMA IR.84 - Video Share Phase 2 Interoperability Specification http://gsmworld.com/documents
[RFC3966]	The tel URI for Telephone Numbers http://tools.ietf.org/html/rfc3966
[ACRDRAFT]	The acr URI for anonymous users http://tools.ietf.org/html/draft-uri-acr-extension-03

4. DOCUMENT MANAGEMENT

Version	Date	Brief Description of Change	Approval Authority	Editor / Company
0.1	28 July 2011	Initial version based on GSMA RCS API requirements document.	GSMA RCE API Tiger Team	Sergio García Murillo Telefónica
0.2	7 Oct 2011	Comments and corrections after RCE API Tiger Team calls RCEAPI001-RCEAPI013 and face-to-face meeting.	GSMA RCE API Tiger Team	Jose M Recio Solaiemes
0.3	10 Oct 2011	Comments and corrections	GSMA RCE API Tiger Team	Sergio García Murillo Telefónica
0.4	13 Oct 2011	File transfer CR	GSMA RCE API Tiger Team	Sergio García Murillo Telefónica
1.0	17 Oct 2011	First candidate version	GSMA RCE API Tiger Team	Sergio García Murillo Telefónica