RCS MaaP Chatbot API Specifications
Version 1.0
27 November 2017

This is a Non-binding Permanent Reference Document of the GSMA

Security Classification: Non-confidential

Access to and distribution of this document is restricted to the persons permitted by the security classification. This document is confidential to the Association and is subject to copyright protection. This document is to be used only for the purposes for which it has been supplied and information contained in it must not be disclosed or in any other way made available, in whole or in part, to persons other than those permitted under the security classification without the prior written approval of the Association.

Copyright Notice

Disclaimer

The GSM Association ("Association") makes no representation, warranty or undertaking (express or implied) with respect to and does not accept any responsibility for, and hereby disclaims liability for the accuracy or completeness or timeliness of the information contained in this document. The information contained in this document may be subject to change without prior notice.

Antitrust Notice

The information contain herein is in full compliance with the GSM Association's antitrust compliance policy.

Table of Contents

1 Introduction

1.1 Overview

This document defines the architecture and a set of standardized Application Programming Interfaces (API) to develop MaaP Chatbots.

1.2 Scope

The scope of this document covers the APIs from Chatbot Platforms / aggregators to the Chatbot developers and brands. The API list covers the common features and it is Chatbot Platform / aggregator responsibility to provide proprietary API functions.

This API set is compliant with the GSMA PRD RCS Universal Profile v2.0 [1]. This API set defines the minimal functional APIs to enable the communications between Chatbots and RCS users. Each Chatbot Platform may provide additional APIs or data properties to meet its own business needs. The OAuth 2.0 authorization framework shall be used to enable a Chatbot to obtain the access to the Chatbot Platform's APIs. The access token scope shall be set as Chatbot message. The communication protocol between the Chatbot Platform and Chatbots shall be HTTPS, in both end points. All date/time fields within JSON payloads should be specified as defined in ISO 8601, including all three fields, date, time, and time zone offset.

This API specification is based on 2 GSMA RCS PRDs:

- RCS Universal Profile v2.0 [1]
  (Main section for Chatbot product manager and designer is Section 15 and Annex A.)
- Rich Communication Suite 7.0 Advanced Communications v8.0 [2]
  (Main section for Chatbot developer is Section 3.6.10)

1.3 Definitions

Term	Description
Aliasing	Conversations with Chatbots are in ‘aliased mode’ as default, i.e. the user’s MSISDN is not used as an identifier for the user but a unique and anonymous ‘token’ or ‘alias’.
Brands / Enterprise	A business or entity that uses messaging to communicate with consumers. Examples include social networks, large and small businesses, financial institutions, schools, medical practices, and non-profits.
Chatbot	An RCS-based service provided to the users whose output is presented in a conversational form and which provides value through brand/enterprise services. Often a piece of software interfacing with one or more users aiming to simulate intelligent human conversation.
Chatbot APIs	A standardised list of a limited set of APIs to facilitate developers and brand in consuming RCS services.
Chatbot Platform	A system that provides a mechanism for Chatbot developers to create and register Chatbots, which can then be exposed to the users connected to the platform through a messaging system.
MaaP	Messaging as a Platform - Enabler layer to enrich communication between businesses (content and service providers) and MNO messaging users

Service Provider	Any company or organisation which allows its subscribers access to the provided services.

1.4 Abbreviations

Term	Description
API	Application Programming Interface
GSMA	GSM Association
MaaP	Messaging as a Platform
MSISDN	Mobile Subscriber Integrated Services Digital Network Number (often referred to the mobile contact number)
RCS	Rich Communication Services
UNI	User-Network Interface

1.5 References

Ref	Doc Number	Title
[1]	[PRD RCC.71]	RCC.71 RCS Universal Profile Service Definition Document v2.0 https://www.gsma.com/futurenetworks/wp-content/uploads/2017/07/RCC.71_v2.0.pdf
[2]	[PRD RCC.07]	Rich Communication Suite 7.0 Advanced Communications Services and Client Specification v8.0 https://www.gsma.com/futurenetworks/wp-content/uploads/2017/06/RCS_7_UNI.zip

1.6 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 [PRD RCC.07]."

2 Models

2.1 RCSMessageWithContactInfo

Description	This is the data model which includes supported RCS message type and contact information.		
Property	**Type**	**Mandatory**	**Description**
RCSMessage	RCSMessage	Yes	A valid RCS message.
messageContact	MessageContact	Yes	A contact presenting a RCS user.

2.2 RCSMessage

Description		This is the data model of a valid RCS message		
Property		**Type**	**Mandatory**	**Description**
anyOf ->	$ref	RCSMessageBase	No	The basic properties about a RCS message exchanged between the user and the Chatbot.

	$ref	RCSMessageType	No	The different RCS message types which can be exchanged between the Chatbot and users.

2.3 RCSMessageBase

Description	This is the data model which includes the basic properties about a RCS message exchanged between the user and the Chatbot.		
Property	**Type**	**Mandatory**	**Description**
msgId	string	No	This is the identifier of the message, and this value is provided by the Chatbot Platform only.
status	MessageStatus	No	This is the status of the message, and this applies to message sent from Chatbot to users and message sent from users to Chatbot.
trafficType	string enum: [advertisement, payment, premium, subscription]	No	This is traffic type specified in US15-6 of RCC.71 [1] and 3.6.7.2 of RCC.07 [2]. The Chatbot should set this value if the traffic belongs one of required type based on the agreement between the Chatbot Platform and the Chatbot.
expiry	string ($date-time)	No	This is the expiry of the message. The Chatbot Platform will try to revoke this message after this expiry.
timestamp	string ($date-time)	No	This is the last date-time the message is updated.

2.4 RCSMessageType

Description		This defines different RCS message types which can be exchanged between the Chatbot and users.		
Property		**Type**	**Mandatory**	**Description**
oneOf ->	suggestedResponse	object	No	This is the Suggested Response JSON object following UP 2.0 specification, a.k.a. the "response" object defined in 3.6.10.3 of RCC.07 [2]. This can only be sent from the user to the Chatbot.
	sharedData	object	No	This is the Shared Data JSON object following UP 2.0 specification, a.k.a. the "sharedData" object defined in 3.6.10.3 of RCC.07 [2]. This can only be sent from the user to the Chatbot.
	isTyping	string	No	The isTyping notification to be sent to the user for the given Chat, or the isTyping notification received from the user for the given Chat. If sending the isTyping notification to the user, the value can be set to 'active' or 'idle'. According to

				RFC3994, the default active-state refresh interval is 120 seconds, and the default idle time-out interval is 15 seconds. In other words, if no 'active' notification is sent by the bot within 15 seconds, the platform will send the 'idle' notification to the user.
	$ref	RCSContentMessageWithSuggestedChipList	No	The RCS message content with suggested chip list.

2.5 RCSContentMessageWithSuggestedChipList

Description		One and only one suggested chip list can be used together with one and only one of textMessage, fileMessage, audioMessage, geolocationPushMessage, or richcardMessage.		
Property		**Type**	**Mandatory**	**Description**
allOf ->	suggestedChipList	object	No	This is the Suggested Chip List JSON object following UP 2.0 specification, a.k.a. the "suggestions" object defined in 3.6.10.3 of RCC.07 [2]. The Chatbot shall not send a chip list alone and it has to be associated with a RCSContentMessage.
	$ref	RCSContentMessage	No	The RCS message content.

2.6 RCSContentMessage

Description		One and only one of textMessage, fileMessage, audioMessage, geolocationPushMessage, or richcardMessage should be provided if sending a message to the user. The user can send textMessage, fileMessage, audioMessage, or geoLocationPushMessage to the Chatbot.		
Property		**Type**	**Mandatory**	**Description**
one of->	textMessage	string	No	This is a normal RCS text based Chat message defined in 3.2.3 of RCC.07 [2].
	fileMessage	FileMessage	No	The RCS file transfer.
	audioMessage	AudioMessage	No	The RCS audio message.
	geolocationPushMessage	GeolocationPushMessage	No	The RCS geolocation push.
	richcardMessage	object	No	This is the Rich Card JSON object following UP 2.0 specification, a.k.a. the "message" object defined in 3.6.10.3 of RCC.07 [2].

2.7 FileMessage

Description	file to be sent via RCS File Transfer defined in 3.2.5 of RCC.07 [2]		
Property	**Type**	**Mandatory**	**Description**
thumbnailFileName	string	No	The file name of the thumbnail.

thumbnailUrl	string ($url)	No	The URL of the thumbnail.
thumbnailMIMEType	string	No	The MIME type of the thumbnail.
thumbnailFileSize	integer	No	The size of the thumbnail.
fileName	string	No	The file name.
fileUrl	string ($url)	Yes	The URL of the file.
fileMIMEType	string	No	The MIME type of the file.
fileSize	integer	No	The size of the file.

2.8 AudioMessage

Description	audio file to be sent via RCS Audio Message defined in 3.2.7 of RCC.07 [2].		
Property	**Type**	**Mandatory**	**Description**
fileName	string	No	The file name.
fileUrl	string ($url)	Yes	The URL of the file.
fileMIMEType	string	No	The MIME type of the file.
fileSize	integer	No	The size of the file.
playingLength	integer minimum: 1 maximum: 600	No	The playing length of the audio.

2.9 GeolocationPushMessage

Description	This is a geolocation push to be sent via RCS Geolocation Push defined in 3.2.6 of RCC.07 [2].		
Property	**Type**	**Mandatory**	**Description**
label	string maxLength: 200 example: meeting location	No	This can be used to tag the nature of the location.
timestamp	string($date-time)	No	This is the time when the location information was pushed.
expiry	string($date-time)	No	This is an absolute date at which time the recipient is no longer permitted to possess the location information.
timeOffset	integer example: -300	No	This is the time zone where the location information was pushed, expressed as the number of minutes away from UTC as defined in [RFC4480].
pos	string example: 26.1181289 -80.1283921	Yes	This is the coordinates in WGS 84 (latitude, longitude) decimal notion as described in [RFC5491], providing the latitude and longitude as “double”-encoded decimal numbers (as specified in [GML3.1.1]) representing the degrees, separated by a space starting with the latitude.

radius	number example: 10	No	The radius of the circle will be represented in meters, which will be indicated by setting the unit of measure attribute of the radius element to the value of EPSG9001 as described in [RFC5491].

2.10 MessageContact

Description		A contact presenting a RCS user.		
Property		**Type**	**Mandatory**	**Description**
any of->	userContact	string	No	Based on the current RCS UP2.0, user contact is MSISDN in E.164 format. Used when the user's phone number is available to be used under the user's agreement. In the future, some mobile operators may allow using other format to identify the user, for example, "user@example_operator.com".
	chatId	string	No	In anonymous mode, the user's phone number is not disclosed to the bot, so this is the anonymous token generated by the Alias Function. The Aliasing concept is explained in US15-2 of RCC.71 [1]. Note: According to R15-2-1-5-2 of RCC.71 [1], the user may choose to link the anonymous token with his/her MSISDN, in such a case, both userContact and chatId will be provided when the user takes the action to link the anonymous token with the MSISDN.

2.11 MessageStatus

Description	This is the status of the message, and this applies to message sent from Chatbot to users and message sent from users to Chatbot.		
Property	**Type**	**Mandatory**	**Description**
	string enum: [pending, sent, delivered, displayed, cancelled, revoked, failed]		• 'pending' - the message status is unknown; • 'sent' - the message has been sent to the contact; • 'delivered' - the message has been successfully delivered to the contact; • 'displayed' - the contact has opened / displayed the message (which probably means they have read it too); • 'canceled' - the message has been requested to be revoked by the sender; • 'revoked' - the message has been revoked successfully;

			• 'failed' - fail to send the message.

2.12 FileData

Description		This is the data model for file upload. Either one of fileUrl or fileContent shall be provided.		
Property		**Type**	**Mandatory**	**Description**
fileType		string	No	This is the MIME content type of the uploaded file. e.g., image/jpeg, audio/mp4, video/mpeg.
until		string ($date-time)	No	This specifies how long the Chatbot wants to keep the file in the CDN.
oneOf ->	fileContent	string ($binary)	No	This is the binary data of the actual file.
	fileUrl	string ($url)	No	This is the url link to a file hosted at somewhere.

2.13 File

Description	This is the file uploaded to the Chatbot Platform CDN for future usage, e.g., file transfer, rich card, audio message, etc.		
Property	**Type**	**Mandatory**	**Description**
fileId	string example: MzJmajlmamVzZGZ8bmk5MHNlbmRmZTAz	Yes	This is the identifier for the uploaded file.
fileUrl	string ($url)	No	This is the url link that the Chatbot can use for file transfer, rich card, audio message, etc.
fileSize	integer($int32)	No	The file size.
status	string enum: [pending, ready, expired, invalid]	No	This is the status of the uploaded file. 'pending'-the file is not ready to use yet; 'ready'-the file is ready to use; 'expired'-the validity expires; 'invalid'-the file cannot be used for some reasons.
validity	string ($date-time)	No	This is the validity of the file determined by the Chatbot Platform and the file may be not accessible after this.

2.14 Reason

Description	This is the data model for the Chatbot Platform to provide additional information regarding the HTTP request and response.		
Property	**Type**	**Mandatory**	**Description**
code	integer($int32)	No	The reason code.
text	string	No	The text description of the given reason.

2.15 WebhookPayload

Description				This is the callback payload the Chatbot will received from the Chatbot Platform.		
Property				**Type**	**Mandatory**	**Description**
allOf ->	**anyOf -> **	**oneOf ->**	RCSMessage	RCSMessage	No	A valid RCS message. Note: The suggestedChipList and richcardMessage are not sent by the user as defined in RCC.71 [1]
			file	File	No	A uploaded file.
		messageContact		MessageContact	No	A contact presenting a RCS user.
		reason		Reason	No	Additional information, if any, from the Chatbot Platform.
	event			string enum: [message, isTyping, messageStatus, fileStatus, response, alias, newUser]	Yes	Event type of the callback: • 'message' - message (text, file, audio, geolocation, etc.) from user; • 'isTyping' - isTyping indication from user; • 'messageStatus' - message status updates, including deliver/display notification and other message status updates; • 'fileStatus' - status updates for the uploaded file; • 'response' - user responses for the suggested replies and suggested actions; • 'alias' - aliasing related events e.g., linking alias token with user's phone number; • 'newUser' - the first time user tries to contact the Chatbot.

3 API

3.1 Resource: messages in a 1-1 chat between Chatbot and user

The resource used is:

```
https://{serverRoot}/bot/{apiVersion}/{botId}/messages
```

3.1.1 Request URL parameters

The following request URL variables are common for all HTTP methods:

Name	Type	Description
serverRoot	path	Server base url: hostname+port+base path Port and base path are OPTIONAL.
apiVersion	path	Version of the API clients want to use. For this specification, it is v1.
botId	path	This is the identifier (client_id) provided by the Chatbot Platform used in OAuth.

3.1.2 POST

This is the API used to send messages and isTyping indications to users.

RCS UP2.0 supports various types of messages that can be sent to users, including text message, file, audio message, geolocation push, rich card, and suggested chip list. However, it is up to the Chatbot Platform or Operator to decide which message type can be actually supported by the underneath network as specified in R15-7-1-1 of RCC.71 [1]. The Chatbot Platform shall always request delivery report and display report for messages sent to users. The Chatbot Platform shall update the message status via the webhook callback provided by the Chatbot.

3.1.2.1 Request

One and only one textMessage, fileMessage, audioMessage, geolocationPushMessage, or richcardMessage can be used if sending messages to users. One and only one suggestedChipList can be used together with the message being sent. The Chatbot shall provide only either userContact or chatId depending on which one is available to the Chatbot. The Chatbot can send isTyping indication to users by setting isTyping to 'active', and it shall continue setting isTyping to 'active' every 15 seconds otherwise the Chatbot Platform would dismiss the isTyping indication to users. The Chatbot can also dismiss the isTyping indication earlier by setting isTyping to 'idle'.

3.1.2.1.1 Example request (informative)

```
curl -X POST -H "Authorization: Bearer ACCESS_TOKEN" -H "Content-Type:
application/JSON"
https://botplatform.example.com/bot/v1/309JF3JSIJFEISIFJOE/messages
```

3.1.2.1.2 Request body schema

Property	Type	Mandatory	Description
$ref	RCSMessageWithContactInfo	Yes	A valid RCS message with contact information

3.1.2.1.3 Example 1: send text message (informative)

```
{
  "RCSMessage": {
    "textMessage": "hello world"
  },
  "messageContact": {
    "userContact": "+14251234567"
  }
}
```

3.1.2.1.4 Example 2: send isTyping indication (informative)

```
{
  "RCSMessage": {
    "isTyping": "active"
  },
  "messageContact": {
    "userContact": "+14251234567"
  }
}
```

3.1.2.1.5 Example 3: send a Rich Card as advertisement with Suggested Chip List (informative)

```
{
  "RCSMessage": {
    "trafficType": "advertisement",
    "richcardMessage": {
      "message": {
        "generalPurposeCard": {
          "layout": {
            "cardOrientation": "HORIZONTAL",
            "imageAlignment": "LEFT"
          },
          "content": {
            "media": {
              "mediaUrl": "https://cdn.server/path/media.mp4",
              "mediaContentType": "video/mp4",
              "mediaFileSize": 2718288,
              "thumbnailUrl": "https://cdn.server/path/media.png",
              "thumbnailContentType": "image/png",
              "thumbnailFileSize": 314159,
              "height": "MEDIUM_HEIGHT",
              "contentDescription": "Textual description of media content,
e. g. for use with screen readers."
            },
            "title": "This is a single rich card.",
            "description": "This is the description of the rich card. It's
the first field that will be truncated if it exceeds the maximum width or
height of a card."
          }
        }
      }
    },
```

```
    "suggestedChipList": {
      "suggestions": [
        {
          "reply": {
            "displayText": "Yes",
            "postback": {
              "data": "set_by_chatbot_reply_yes"
            }
          }
        },
        {
          "reply": {
            "displayText": "No",
            "postback": {
              "data": "set_by_chatbot_reply_no"
            }
          }
        },
        {
          "action": {
            "urlAction": {
              "openUrl": {
                "url": "https://www.gsma.com"
              }
            },
            "displayText": "Open website or deep link",
            "postback": {
              "data": "set_by_chatbot_open_url"
            }
          }
        }
      ]
    }
  },
  "messageContact": {
    "userContact": "+14251234567"
  }
}
```

3.1.2.2 Responses

Code	Description
202	The request of sending message or isTyping indication is accepted by the Chatbot Platform and ready to send to the user.
400	This is a bad request with invalid input, invalid object, etc.
401	This request is unauthorized.
404	The user contact or the chat ID cannot be found.
5XX	Server error.

3.1.2.2.1 Response body schema

Property		Type	Mandatory	Description
RCSMessage	msgId	string	Yes	This is the identifier generated by the Chatbot platform for the message/isTyping to be sent.
	status	string enum: [pending, sent, failed]	Yes	This is the initial status of the message/isTyping to be sent. 'pending' means it may take further process before the message can be sent to the user; 'sent' means the message has been sent to the user via operator's network; 'failed' means the message cannot be sent to the user via operatior's network. The Chatbot Platform shall provide further update of the message delivery status via webhook.
	timestamp	string ($date-time)	No	This is the date-time the message sent request is accepted.
reason		Reason	No	Additional information, if any, from the Chatbot Platform.

3.1.2.2.2 Example 1: sending message requested has been accepted (informative)

```
{
  "RCSMessage": {
    "msgId": "MzJmajlmamVzZGZ8bmk5MHNlbmRmZTAz",
    "status": "pending"
  }
}
```

3.2 Resource: message status

The resource used is:

`https://{serverRoot}/bot/{apiVersion}/{botId}/messages/{msdId}/status`

3.2.1 Request URL parameters

The following request URL variables are common for all HTTP methods:

Name	Type	Description
serverRoot	path	Server base url: hostname+port+base path Port and base path are OPTIONAL.
apiVersion	path	Version of the API clients want to use. For this specification, it is v1.
botId	path	This is the identifier (client_id) provided by the Chatbot Platform used in OAuth.
msgId	path	This is the message identifier.

3.2.2 GET

This is the API to query the given message status.

Although the message status can be updated via webhook, this API provides an alternative optional way to check the message status. Possible message status includes 'pending', 'sent', 'delivered', 'displayed', 'cancelled', 'revoked', and 'failed'.

3.2.2.1 Request

3.2.2.1.1 Example request (informative)

```
curl -X GET -H "Authorization: Bearer ACCESS_TOKEN"
https://botplatform.example.com/bot/v1/309JF3JSIJFEISIFJOE/messages/MzJmajl
mamVzZGZ8bmk5MHNlbmRmZTAz/status
```

3.2.2.2 Responses

Code	Description
200	OK
401	The request is unauthorized.
404	The message ID cannot be found.
5XX	Server error

3.2.2.2.1 Response body schema

Property		Type	Mandatory	Description
RCSMessage	msgId	string	Yes	This is the identifier for the message being queried
	status	MessageStatus	Yes	Possible message status includes 'pending', 'sent', 'delivered', 'displayed', 'cancelled', 'revoked', and 'failed'.
	timestamp	string ($date-time)	No	This is the last date-time the message is updated.
reason		Reason	No	Additional information, if any, from the Chatbot Platform.

3.2.2.2.2 Example 1: query message status (informative)

```
{
  "RCSMessage": {
    "msgId": "MzJmajlmamVzZGZ8bmk5MHNlbmRmZTAz",
    "status": "delivered",
    "timestamp": "2017-10-19T22:20:49.718Z"
  }
}
```

3.2.3 PUT

This is the API to send read notification to users for a received message or revoke a sent message.

Message status that can be updated includes 'displayed' and 'cancelled'. When the status is marked as 'displayed', the Chatbot Platform shall send a display notification to the user if it has not been sent before. When the status is marked as 'cancelled', the Chatbot Platform shall try to revoke the message if it has not been delivered to the user. If the message revocation is successful, the Chatbot Platform shall notify the Chatbot via webhook by marking the status as 'revoked'.

3.2.3.1 Request

3.2.3.1.1 Example request (informative)

```
curl -X PUT -H "Authorization: Bearer ACCESS_TOKEN" -H "Content-Type:
application/JSON"
https://botplatform.example.com/bot/v1/309JF3JSIJFEISIFJOE/messages/MzJmajl
mamVzZGZ8bmk5MHNlbmRmZTAz/status
```

3.2.3.1.2 Request body schema

Property		Type	Mandatory	Description
RCSMessage	status	string enum: [displayed, cancelled]	Yes	The message status that the Chatbot wants to set to.

3.2.3.1.3 Example 1: sent displayed notification (informative)

```
{
  "RCSMessage": {
    "status": "displayed"
  }
}
```

3.2.3.2 Responses

Code	Description
204	The status of the message has been updated by the Chatbot Platform. For 'displayed', a display notification will be sent to the user; for 'cancelled', the Chatbot Platform shall try to revoke the message if it has not been delivered to the user. NOTE, how to revoke a message is still under GSMA discussion so the Chatbot Platform may or may not support this operation.
400	This is a bad request with invalid input, invalid object, etc.
401	The request is unauthorized.
404	The message ID cannot be found.
5XX	Server error

3.2.3.2.1 Response body schema

Property	Type	Mandatory	Description
reason	Reason	No	Additional information, if any, from the Chatbot Platform.

3.3 Resource: remote contact capability

The resource used is:

```
https://{serverRoot}/bot/{apiVersion}/{botId}/contactCapabilities
```

3.3.1 Request URL parameters

The following request URL variables are common for all HTTP methods:

Name	Type	Description
serverRoot	path	Server base url: hostname+port+base path Port and base path are OPTIONAL.
apiVersion	path	Version of the API clients want to use. For this specification, it is v1.
botId	path	This is the identifier (client_id) provided by the Chatbot Platform used in OAuth.
userContact	query	Based on the current RCS UP2.0, user contact is MSISDN in E.164 format, for example, "+14251234567". Used when the user's phone number is available to be used under the user's agreement. In the future, some mobile operators may allow using other format to identify the user, for example, "user@example_operator.com".
chatId	query	In anonymous mode, the user's phone number is not disclosed to the Chatbot, so an anonymous token is generated by the Alias Function. This is the remote contact presented as chatId due to aliasing. The Aliasing concept is explained in US15-2 of RCC.71.

3.3.2 GET

This is the API to get the RCS capability of the given user's device.

Because this is a RCS based communication service, the Chatbot shall only communicate with the user using a RCS capable device. So the Chatbot shall conduct the RCS capability discovery to learn about whether the given user's device is RCS capable or not. When the RCS capability discovery shall be conducted is based on the policy of the Chatbot Platform or Operator, and it is out of the scope of this API specification. Possible capabilities include

- 'chat' (text message),
- 'fileTransfer' (standalone file transfer and AMR audio message),
- 'videoCall' (video calling),
- 'geolocationPush' (geolocation information),

- 'callComposer' (enrich calling pre-call setup),
- 'chatBotCommunication' (Rich Card and Suggested Chip List). The Chatbot shall only send the message type that the contact can support. The Chatbot can only send Rich Card and Suggested Chip List to the contact who has the 'chatBotCommunication' capability. To query the capability, the Chatbot shall provide only either userContact or chatId depending on which one is available to the Chatbot.

3.3.2.1 Request

3.3.2.1.1 Example request (informative)

```
curl -X GET -H "Authorization: Bearer ACCESS_TOKEN"
https://botplatform.example.com/bot/v1/309JF3JSIJFEISIFJOE/contactCapabilit
ies?userContact=%2B14251234567
```

3.3.2.2 Responses

Code	Description
200	OK
401	The request is unauthorized.
404	The user contact or chat ID cannot be found or the given user's device is not RCS capabile.
5XX	Server error

3.3.2.2.1 Response body schema

Property	Type	Mandatory	Description
capabilities	string array enum: [chat, fileTransfer, videoCall, geolocationPush, callComposer, chatBotCommunication]	No	The capabilities list
reason	Reason	No	Additional information, if any, from the Chatbot Platform.

3.3.2.2.2 Example 1: query user capabilities

```
{
  "capabilities": [
    "chatBotCommunication",”fileTransfer”
  ]
}
```

3.4 Resources: files uploaded to Chatbot Platform CDN

This is not for file transfer, but uploading files from Chatbot to the Chatbot platform CDN.

The resources used are:

```
https://{serverRoot}/bot/{apiVersion}/{botId}/files
https://{serverRoot}/bot/{apiVersion}/{botId}/files/{fileId}
```

3.4.1 Request URL parameters

The following request URL variables are common for all HTTP methods:

Name	Type	Description
serverRoot	path	Server base url: hostname+port+base path Port and base path are OPTIONAL.
apiVersion	path	Version of the API clients want to use. For this specification, it is v1.
botId	path	This is the identifier (client_id) provided by the Chatbot Platform used in OAuth.
fileId	path	This is the file identifier.

3.4.2 POST

This is the API to upload a file to the CDN of the Chatbot Platform.

Based on the policy, the Chatbot Platform may require each file to be sent to users to be uploaded to its CDN first. The uploaded file can be used later as media content in the rich card or other message types such as file transfer or audio message. The file can be uploaded directly or by sharing a file URL.

3.4.2.1 Request

The request content type is multipart/form-data.

3.4.2.1.1 Request body schema

Property	Type	Mandatory	Description
$ref	FileData	Yes	Information about a file to be uploaded.

3.4.2.1.2 Example 1: upload a raw file (informative)

```
POST /123456/files
HOST: example.com
Content-Length: xxx
Content-Type: multipart/form-data; boundary=----
WebKitFormBoundaryWfPNVh4wuWBlyEyQ

------WebKitFormBoundaryWfPNVh4wuWBlyEyQ
Content-Disposition: form-data; name="fileType"

audio/mp4
------WebKitFormBoundaryWfPNVh4wuWBlyEyQ
Content-Disposition: form-data; name="until"

2017-10-03T21:08:15.933Z
------WebKitFormBoundaryWfPNVh4wuWBlyEyQ
```

```
Content-Disposition: form-data; name="fileContent"; filename="audio.mp4"
Content-Type: audio/mp4

[file content goes there]
------WebKitFormBoundaryWfPNVh4wuWBlyEyQ
```

3.4.2.1.3 Example 2: upload a file by URL (informative)

```
POST /123456/files
HOST: example.com
Content-Length: xxx
Content-Type: multipart/form-data; boundary=----
WebKitFormBoundaryWfPNVh4wuWBlyEyQ

------WebKitFormBoundaryWfPNVh4wuWBlyEyQ
Content-Disposition: form-data; name="fileType"

audio/mp4
------WebKitFormBoundaryWfPNVh4wuWBlyEyQ
Content-Disposition: form-data; name="until"

2017-10-03T21:08:15.933Z
------WebKitFormBoundaryWfPNVh4wuWBlyEyQ
Content-Disposition: form-data; name="fileUrl"

http://www.example.com/files/example-audio.mp4
------WebKitFormBoundaryWfPNVh4wuWBlyEyQ
```

3.4.2.2 Responses

Code	Description
202	The file upload request has been accepted. The Chatbot Platform will notify the Chatbot, via webhook, whether the file is ready for use in the communication with RCS users.
400	This is a bad request with invalid input, invalid object, etc.
401	The request is unauthorized.
5XX	Server error

3.4.2.2.1 Response body schema

Property	Type	Mandatory	Description
file	File	No	Information about the uploaded file.
reason	Reason	No	Additional information, if any, from the Chatbot Platform.

3.4.2.2.2 Example 1: upload a file response (informative)

```
{
  "file": {
    "fileId": "MzJmajlmamVzZGZ8bmk5MHNlbmRmZTAz",
    "fileUrl": "string",
```

```
    "fileSize": 56000,
    "status": "pending",
    "validity": "2017-10-19T23:31:34.136Z"
  }
}
```

3.4.3 DELETE

This is the API to delete the file which was previously uploaded to the CDN of the Chatbot Platform.

Once the file is deleted, it won't be available for use any more. In addition, any reference to this file would be invalid.

3.4.3.1 Request

3.4.3.1.1 Example request (informative)

```
curl -X DELETE -H "Authorization: Bearer ACCESS_TOKEN"
https://botplatform.example.com/bot/v1/xFX3ijI4TeiqrE3QyCMHrw/files/bot-
temp-file-upload-866f331d-d1cb-489c-8154-233662b98dd5
```

3.4.3.2 Responses

Code	Description
204	The file has been deleted
401	The request is unauthorized
404	The file cannot be found
5XX	Server error

3.4.3.2.1 Response body schema

Property	Type	Mandatory	Description
reason	Reason	No	Additional information, if any, from the Chatbot Platform.

3.4.4 GET

This is the API to retrieve a file's information.

Although the file status can be updated via webhook, this API provides an alternative optional way to check the file status along with other information. Possible file status includes 'pending', 'ready', 'expired', and 'invalid'.

3.4.4.1 Request

3.4.4.1.1 Example request (informative)

```
curl -X GET -H "Authorization: Bearer ACCESS_TOKEN"
https://botplatform.example.com/bot/v1/xFX3ijI4TeiqrE3QyCMHrw/files/bot-
temp-file-upload-866f331d-d1cb-489c-8154-233662b98dd5
```

3.4.4.2 Responses

Code	Description
200	OK
401	The request is unauthorized
404	The file cannot be found

3.4.4.2.1 Response body schema

Property	Type	Mandatory	Description
file	File	No	Information about the uploaded file.
reason	Reason	No	Additional information, if any, from the Chatbot Platform.

3.5 Webhook

RCS event from the Chatbot Platform to Chatbot. Webhook is provided by the Chatbot.

3.5.1 POST

This is the callback API exposed by the Chatbot from which the Chatbot Platform can send information to the Chatbot.

The Chatbot Platform uses webhook exposed by Chatbot to send a HTTP POST payload when certain RCS events occur. How the Chatbot Platform configures webhooks with Chatbots is out of the scope of this specification. Chatbot Platform may send following events to Chatbot:

1. message from user;
2. isTyping notification from user;
3. message status update;
4. uploaded file status update;
5. response to suggested reply or action;
6. alias link command;
7. new user.

The Chatbot shall always return a 200 OK HTTP response to the HTTP POST from the Chatbot Platform.

3.5.1.1 Request

3.5.1.1.1 Request body schema

Property	Type	Mandatory	Description
$ref	WebhookPayload	Yes	RCS event posted to webhook by Chatbot Platform

3.5.1.1.2 Example 1: receive text message (informative)

```
{
  "RCSMessage": {
```

```
    "msgId": "Xs8CI3tdf",
    "textMessage": "hello world",
    "timestamp": "2017-09-26T01:33:20.315Z"
  },
  "messageContact": {
    "userContact": "+14251234567"
  },
  "event": "message"
}
```

3.5.1.1.3 Example 2: receive text message from a user in aliasing mode (informative)

```
{
  "RCSMessage": {
    "msgId": "Xs8CI3tdf",
    "textMessage": "hello world",
    "timestamp": "2017-09-26T01:33:20.315Z"
  },
  "messageContact": {
    "chatId": "93JF93SEIJFE"
  },
  "event": "message"
}
```

3.5.1.1.4 Example 3: receive a geolocation push (informative)

```
{
  "RCSMessage": {
    "msgId": "Xs8CI3tdf",
    "geolocationPushMessage": {
      "label": "meeting location",
      "timestamp": "2017-09-26T01:46:04.868Z",
      "expiry": "2017-09-26T01:46:04.868Z",
      "timeOffset": -300,
      "pos": "26.1181289 -80.1283921",
      "radius": 10
    },
    "timestamp": "2017-09-26T01:33:20.315Z"
  },
  "messageContact": {
    "userContact": "+14251234567"
  },
  "event": "message"
}
```

3.5.1.1.5 Example 4: receive a file (informative)

```
{
  "RCSMessage": {
    "msgId": "Xs8CI3tdf",
    "fileMessage": {
      "thumbnailFileName": "t.jpg",
      "thumbnailUrl": "http://www.example.com/files/t.jpg",
      "thumbnailMIMEType": "image/jpeg",
```

```
      "thumbnailFileSize": 1234,
      "fileName": "f.jpg",
      "fileUrl": "http://www.example.com/files/f.jpg",
      "fileMIMEType": "image/jpeg",
      "fileSize": 1234567
    },
    "timestamp": "2017-09-26T01:33:20.315Z"
  },
  "messageContact": {
    "userContact": "+14251234567"
  },
  "event": "message"
}
```

3.5.1.1.6 Example 5: receive isTyping indication (informative)

```
{
  "RCSMessage": {
    "msgId": "Xs8CI3tdf",
    "isTyping": "active",
    "timestamp": "2017-09-26T01:33:20.315Z"
  },
  "messageContact": {
    "userContact": "+14251234567"
  },
  "event": "isTyping"
}
```

3.5.1.1.7 Example 6: receive message read notification from the user (informative)

```
{
  "RCSMessage": {
    "msgId": "MzJmajlmamVzZGZ8bmk5MHNlbmRmZTAz"
    "status": "displayed",
    "timestamp": "2017-09-26T01:33:20.315Z"
  },
  "messageContact": {
    "userContact": "+14251234567"
  },
  "event": "messageStatus"
}
```

3.5.1.1.8 Example 7: receive message send failure notification (informative)

```
{
  "RCSMessage": {
    "msgId": "MzJmajlmamVzZGZ8bmk5MHNlbmRmZTAz"
    "status": "failed",
    "timestamp": "2017-09-26T01:33:20.315Z"
  },
  "event": "messageStatus"
}
```

3.5.1.1.9 Example 8: receive file upload notification (informative)

```
{
```

```
  "file": {
    "fileId": "MzJmajlmamVzZGZ8bmk5MHNlbmRmZTAz",
    "fileUrl": "http://www.example.com/files/f.jpg",
    "fileSize": 123456,
    "status": "ready",
    "validity": "2017-10-03T22:31:00.597Z"
  },
  "event": "fileStatus"
}
```

3.5.1.1.10 Example 9: receive response to a suggested reply the user selects (informative)

```
{
  "RCSMessage": {
    "msgId": "MzJmajlmamVzZGZ8bmk5MHNlbmRmZTAz",
    "suggestedResponse": {
      "response": {
        "reply": {
          "displayText": "Yes",
          "postback": {
            "data": "set_by_chatbot_reply_yes"
          }
        }
      }
    },
    "timestamp": "2017-09-26T01:33:20.315Z"
  },
  "messageContact": {
    "userContact": "+14251234567"
  },
  "event": "response"
}
```

3.5.1.1.11 Example 10: receive postback to a suggested action the user takes (informative)

```
{
  "RCSMessage": {
    "msgId": "MzJmajlmamVzZGZ8bmk5MHNlbmRmZTAz",
    "suggestedResponse": {
      "response": {
        "action": {
          "displayText": "Visit Website",
          "postback": {
            "data": "set_by_chatbot_reply_yes"
          }
        }
      }
    },
    "timestamp": "2017-09-26T01:33:20.315Z"
  },
  "messageContact": {
    "userContact": "+14251234567"
```

```
  },
  "event": "response"
}
```

3.5.1.1.12 Example 11: receive the event (special postback) that a new user wants to chat with the bot (informative)

```
{
  "RCSMessage": {
    "msgId": "MzJmajlmamVzZGZ8bmk5MHNlbmRmZTAz",
    "suggestedResponse": {
      "response": {
        "reply": {
          "displayText": "Start Chat",
          "postback": {
            "data": "new_bot_user_initiation"
          }
        }
      }
    },
    "timestamp": "2017-09-26T01:33:20.315Z"
  },
  "messageContact": {
    "userContact": "+14251234567"
  },
  "event": "newUser"
}
```

3.5.1.1.13 Example 12: receive the event of linking chatId with userContact (R15-2-1-5-2 of RCC.71) (informative)

```
{
  "RCSMessage": {
    "msgId": "MzJmajlmamVzZGZ8bmk5MHNlbmRmZTAz",
    "timestamp": "2017-09-26T01:33:20.315Z"
  },
  "messageContact": {
    "userContact": "+14251234567",
    "chatId": "93JF93SEIJFE"
  },
  "event": "alias"
}
```

3.5.1.2 Responses

Code	Description
200	The Callback playload is received by the Chatbot.

Annex B Document Management

A.1 Document History

Version	Date	Brief Description of Change	Approval Authority	Editor / Company
1.0	22.11.2017	Initial release	TG	Zhinan Zhou / Samsung

A.2 Other Information

Type	Description
Document Owner	RCS MaaP
Editor / Company	Erdem Ersoz / GSMA

It is our intention to provide a quality product for your use. If you find any errors or omissions, please contact us with your comments. You may notify us at PRD@gsma.com

Your comments or suggestions & questions are always welcome.