## Table of Contents

1 **Introduction** 4
   1.1 Overview 4
   1.2 Scope 4
   1.3 Definitions 4
   1.4 Abbreviations 5
   1.5 References 5
   1.6 Conventions 5

2 **Models** 5
   2.1 RCSMessageWithContactInfo 5
   2.2 RCSMessage 5
   2.3 RCSMessageBase 6
   2.4 RCSMessageType 6
   2.5 RCSContentMessageWithSuggestedChipList 7
   2.6 RCSContentMessage 7
   2.7 FileMessage 7
   2.8 AudioMessage 8
   2.9 GeolocationPushMessage 8
   2.10 MessageContact 9
   2.11 MessageStatus 9
   2.12 FileData 10
   2.13 File 10
   2.14 Reason 10
   2.15 WebhookPayload 11

3 **API** 11
   3.1 Resource: messages in a 1-1 chat between Chatbot and user 11
      3.1.1 Request URL parameters 12
      3.1.2 POST 12
   3.2 Resource: message status 15
      3.2.1 Request URL parameters 15
      3.2.2 GET 16
      3.2.3 PUT 16
   3.3 Resource: remote contact capability 18
      3.3.1 Request URL parameters 18
      3.3.2 GET 18
   3.4 Resources: files uploaded to Chatbot Platform CDN 19
      3.4.1 Request URL parameters 20
      3.4.2 POST 20
      3.4.3 DELETE 22
      3.4.4 GET 22
   3.5 Webhook 23
      3.5.1 POST 23

Annex B **Document Management** 28
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

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliasing</td>
<td>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’.</td>
</tr>
<tr>
<td>Brands / Enterprise</td>
<td>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.</td>
</tr>
<tr>
<td>Chatbot</td>
<td>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.</td>
</tr>
<tr>
<td>Chatbot APIs</td>
<td>A standardised list of a limited set of APIs to facilitate developers and brand in consuming RCS services.</td>
</tr>
<tr>
<td>Chatbot Platform</td>
<td>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.</td>
</tr>
<tr>
<td>MaaP</td>
<td>Messaging as a Platform - Enabler layer to enrich communication between businesses (content and service providers) and MNO messaging users.</td>
</tr>
</tbody>
</table>
Service Provider | Any company or organisation which allows its subscribers access to the provided services.
---|---

### 1.4 Abbreviations

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

### 1.5 References

<table>
<thead>
<tr>
<th>Ref</th>
<th>Doc Number</th>
<th>Title</th>
</tr>
</thead>
</table>

### 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

This is the data model which includes supported RCS message type and contact information.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Mandatory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RCSMessage</td>
<td>RCSMessage</td>
<td>Yes</td>
<td>A valid RCS message.</td>
</tr>
<tr>
<td>messageContact</td>
<td>MessageContact</td>
<td>Yes</td>
<td>A contact presenting a RCS user.</td>
</tr>
</tbody>
</table>

#### 2.2 RCSMessage

This is the data model of a valid RCS message

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Mandatory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>anyOf $ref</td>
<td>$ref</td>
<td>No</td>
<td>The basic properties about a RCS message exchanged between the user and the Chatbot.</td>
</tr>
</tbody>
</table>
### 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.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Mandatory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msgId</td>
<td>string</td>
<td>No</td>
<td>This is the identifier of the message, and this value is provided by the Chatbot Platform only.</td>
</tr>
<tr>
<td>status</td>
<td>MessageStatus</td>
<td>No</td>
<td>This is the status of the message, and this applies to message sent from Chatbot to users and message sent from users to Chatbot.</td>
</tr>
<tr>
<td>trafficType</td>
<td>string (enum: [advertisement, payment, premium, subscription])</td>
<td>No</td>
<td>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.</td>
</tr>
<tr>
<td>expiry</td>
<td>string ($date-time)</td>
<td>No</td>
<td>This is the expiry of the message. The Chatbot Platform will try to revoke this message after this expiry.</td>
</tr>
<tr>
<td>timestamp</td>
<td>string ($date-time)</td>
<td>No</td>
<td>This is the last date-time the message is updated.</td>
</tr>
</tbody>
</table>

### 2.4 RCSMessageType

**Description**

This defines different RCS message types which can be exchanged between the Chatbot and users.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Mandatory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>oneOf</td>
<td>suggestedResponse</td>
<td>No</td>
<td>This is the Suggested Response JSON object following UP 2.0 specification, a.k.a. the &quot;response&quot; object defined in 3.6.10.3 of RCC.07 [2]. This can only be sent from the user to the Chatbot.</td>
</tr>
<tr>
<td>sharedData</td>
<td>object</td>
<td>No</td>
<td>This is the Shared Data JSON object following UP 2.0 specification, a.k.a. the &quot;sharedData&quot; object defined in 3.6.10.3 of RCC.07 [2]. This can only be sent from the user to the Chatbot.</td>
</tr>
<tr>
<td>isTyping</td>
<td>string</td>
<td>No</td>
<td>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</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Mandatory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>allOf</td>
<td>suggestedChipList</td>
<td>object</td>
<td>This is the Suggested Chip List JSON object following UP 2.0 specification, a.k.a. the &quot;suggestions&quot; 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.</td>
</tr>
</tbody>
</table>

$ref  
RCSContentMessage

No  The RCS message content.

2.6 RCSContentMessage

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Mandatory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>one of-&gt;</td>
<td>textMessage</td>
<td>string</td>
<td>No  This is a normal RCS text based Chat message defined in 3.2.3 of RCC.07 [2].</td>
</tr>
<tr>
<td></td>
<td>fileMessage</td>
<td>FileMessage</td>
<td>No  The RCS file transfer.</td>
</tr>
<tr>
<td></td>
<td>audioMessage</td>
<td>AudioMessage</td>
<td>No  The RCS audio message.</td>
</tr>
<tr>
<td></td>
<td>geolocationPushMessage</td>
<td>GeolocationPushMessage</td>
<td>No  The RCS geolocation push.</td>
</tr>
<tr>
<td></td>
<td>richcardMessage</td>
<td>object</td>
<td>No  This is the Rich Card JSON object following UP 2.0 specification, a.k.a. the &quot;message&quot; object defined in 3.6.10.3 of RCC.07 [2].</td>
</tr>
</tbody>
</table>

2.7 FileMessage

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Mandatory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>thumbnailFileName</td>
<td>string</td>
<td>No</td>
<td>The file name of the thumbnail.</td>
</tr>
</tbody>
</table>
### thumbnailUrl
- **Type**: string
- **Mandatory**: No
- **Description**: The URL of the thumbnail.

### thumbnailMIMEType
- **Type**: string
- **Mandatory**: No
- **Description**: The MIME type of the thumbnail.

### thumbnailFileSize
- **Type**: integer
- **Mandatory**: No
- **Description**: The size of the thumbnail.

### fileName
- **Type**: string
- **Mandatory**: No
- **Description**: The file name.

### fileUrl
- **Type**: string ($url)
- **Mandatory**: Yes
- **Description**: The URL of the file.

### fileMIMETYPE
- **Type**: string
- **Mandatory**: No
- **Description**: The MIME type of the file.

### fileSize
- **Type**: integer
- **Mandatory**: No
- **Description**: 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].

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Mandatory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileName</td>
<td>string</td>
<td>No</td>
<td>The file name.</td>
</tr>
<tr>
<td>fileUrl</td>
<td>string ($url)</td>
<td>Yes</td>
<td>The URL of the file.</td>
</tr>
<tr>
<td>fileMIMETYPE</td>
<td>string</td>
<td>No</td>
<td>The MIME type of the file.</td>
</tr>
<tr>
<td>fileSize</td>
<td>integer</td>
<td>No</td>
<td>The size of the file.</td>
</tr>
<tr>
<td>playingLength</td>
<td>integer</td>
<td>minimum: 1, maximum: 600</td>
<td>No</td>
</tr>
</tbody>
</table>

### 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].

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Mandatory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>label</td>
<td>string</td>
<td>No</td>
<td>This can be used to tag the nature of the location.</td>
</tr>
<tr>
<td>maxLength: 200 example: meeting location</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>timestamp</td>
<td>string($date-time)</td>
<td>No</td>
<td>This is the time when the location information was pushed.</td>
</tr>
<tr>
<td>expiry</td>
<td>string($date-time)</td>
<td>No</td>
<td>This is an absolute date at which time the recipient is no longer permitted to possess the location information.</td>
</tr>
<tr>
<td>timeOffset</td>
<td>integer</td>
<td>No</td>
<td>This is the time zone where the location information was pushed, expressed as the number of minutes away from UTC as defined in [RFC4480].</td>
</tr>
<tr>
<td>example: -300</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pos</td>
<td>string</td>
<td>Yes</td>
<td>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.</td>
</tr>
<tr>
<td>example: 26.118 1289 - 80.1283921</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Mandatory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>any of -&gt;</td>
<td>userContact</td>
<td>string</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>chatId</td>
<td>string</td>
<td>No</td>
</tr>
</tbody>
</table>

2.11 MessageStatus

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Mandatory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string enum: [ pending, sent, delivered, displayed, cancelled, revoked, failed]</td>
<td></td>
<td></td>
<td>• '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;</td>
</tr>
</tbody>
</table>
2.12 FileData

**Description**
This is the data model for file upload. Either one of fileUrl or fileContent shall be provided.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Mandatory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileType</td>
<td>string</td>
<td>No</td>
<td>This is the MIME content type of the uploaded file. e.g., image/jpeg, audio/mp4, video/mpeg.</td>
</tr>
<tr>
<td>until</td>
<td>string ($date-time)</td>
<td>No</td>
<td>This specifies how long the Chatbot wants to keep the file in the CDN.</td>
</tr>
<tr>
<td>oneOf</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-&gt; fileContent</td>
<td>string ($binary)</td>
<td>No</td>
<td>This is the binary data of the actual file.</td>
</tr>
<tr>
<td>-&gt; fileUrl</td>
<td>string ($url)</td>
<td>No</td>
<td>This is the url link to a file hosted at somewhere.</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Mandatory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileId</td>
<td>string</td>
<td>Yes</td>
<td>This is the identifier for the uploaded file.</td>
</tr>
<tr>
<td>fileUrl</td>
<td>string ($url)</td>
<td>No</td>
<td>This is the url link that the Chatbot can use for file transfer, rich card, audio message, etc.</td>
</tr>
<tr>
<td>fileSize</td>
<td>integer($int32)</td>
<td>No</td>
<td>The file size.</td>
</tr>
<tr>
<td>status</td>
<td>string</td>
<td>No</td>
<td>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.</td>
</tr>
<tr>
<td>validity</td>
<td>string ($date-time)</td>
<td>No</td>
<td>This is the validity of the file determined by the Chatbot Platform and the file may be not accessible after this.</td>
</tr>
</tbody>
</table>

2.14 Reason

**Description**
This is the data model for the Chatbot Platform to provide additional information regarding the HTTP request and response.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Mandatory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td>integer($int32)</td>
<td>No</td>
<td>The reason code.</td>
</tr>
<tr>
<td>text</td>
<td>string</td>
<td>No</td>
<td>The text description of the given reason.</td>
</tr>
</tbody>
</table>
### 2.15 WebhookPayload

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

### 3 API

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

The resource used is:
3.1.1 Request URL parameters
The following request URL variables are common for all HTTP methods:

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

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)
```
```

3.1.2.2 Request body schema

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Mandatory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ref</td>
<td>RCSMessageWithContactInfo</td>
<td>Yes</td>
<td>A valid RCS message with contact information</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>202</td>
<td>The request of sending message or isTyping indication is accepted by the Chatbot Platform and ready to send to the user.</td>
</tr>
<tr>
<td>400</td>
<td>This is a bad request with invalid input, invalid object, etc.</td>
</tr>
<tr>
<td>401</td>
<td>This request is unauthorized.</td>
</tr>
<tr>
<td>404</td>
<td>The user contact or the chat ID cannot be found.</td>
</tr>
<tr>
<td>5XX</td>
<td>Server error.</td>
</tr>
</tbody>
</table>
3.1.2.2.1 Response body schema

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Mandatory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RCSMessage msgId</td>
<td>string</td>
<td>Yes</td>
<td>This is the identifier generated by the Chatbot platform for the message/isTyping to be sent.</td>
</tr>
<tr>
<td>status</td>
<td>string enum: [pending, sent, failed]</td>
<td>Yes</td>
<td>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 operator's network. The Chatbot Platform shall provide further update of the message delivery status via webhook.</td>
</tr>
<tr>
<td>timestamp</td>
<td>string ($date-time)</td>
<td>No</td>
<td>This is the date-time the message sent request is accepted.</td>
</tr>
<tr>
<td>reason</td>
<td>Reason</td>
<td>No</td>
<td>Additional information, if any, from the Chatbot Platform.</td>
</tr>
</tbody>
</table>

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/{msgId}/status

3.2.1 Request URL parameters

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverRoot</td>
<td>path</td>
<td>Server base url: hostname+port+base path</td>
</tr>
<tr>
<td>apiVersion</td>
<td>path</td>
<td>Port and base path are OPTIONAL.</td>
</tr>
<tr>
<td>botId</td>
<td>path</td>
<td>This is the identifier (client_id) provided by the Chatbot Platform used in OAuth.</td>
</tr>
<tr>
<td>msgId</td>
<td>path</td>
<td>This is the message identifier.</td>
</tr>
</tbody>
</table>
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/MzJmajlmamVzZGZ8bmk5MHNlbmRmZTAz/status
```

3.2.2.2 Responses

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td>401</td>
<td>The request is unauthorized.</td>
</tr>
<tr>
<td>404</td>
<td>The message ID cannot be found.</td>
</tr>
<tr>
<td>5XX</td>
<td>Server error</td>
</tr>
</tbody>
</table>

3.2.2.2.1 Response body schema

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

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)

```bash
curl -X PUT -H "Authorization: Bearer ACCESS_TOKEN" -H "Content-Type: application/JSON"
https://botplatform.example.com/bot/v1/309JF3JSIJEISIFJ0E/messages/MzJmajlmamVzGZ8bmk5MHNlbmRmZTAz/status
```

3.2.3.1.2 Request body schema

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Mandatory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RCSMessage</td>
<td>status</td>
<td>string</td>
<td>enum: [displayed, cancelled]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Yes</td>
<td>The message status that the Chatbot wants to set to.</td>
</tr>
</tbody>
</table>

3.2.3.1.3 Example 1: sent displayed notification (informative)

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

3.2.3.2 Responses

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>204</td>
<td>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.</td>
</tr>
<tr>
<td>400</td>
<td>This is a bad request with invalid input, invalid object, etc.</td>
</tr>
<tr>
<td>401</td>
<td>The request is unauthorized.</td>
</tr>
<tr>
<td>404</td>
<td>The message ID cannot be found.</td>
</tr>
<tr>
<td>5XX</td>
<td>Server error</td>
</tr>
</tbody>
</table>

3.2.3.2.1 Response body schema

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Mandatory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reason</td>
<td>Reason</td>
<td>No</td>
<td>Additional information, if any, from the Chatbot Platform.</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverRoot</td>
<td>path</td>
<td>Server base url: hostname+port+base path Port and base path are OPTIONAL.</td>
</tr>
<tr>
<td>apiVersion</td>
<td>path</td>
<td>Version of the API clients want to use. For this specification, it is v1.</td>
</tr>
<tr>
<td>botId</td>
<td>path</td>
<td>This is the identifier (client_id) provided by the Chatbot Platform used in OAuth.</td>
</tr>
<tr>
<td>userContact</td>
<td>query</td>
<td>Based on the current RCS UP2.0, user contact is MSISDN in E.164 format, for example, &quot;+14251234567&quot;. 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, &quot;<a href="mailto:user@example_operator.com">user@example_operator.com</a>&quot;.</td>
</tr>
<tr>
<td>chatId</td>
<td>query</td>
<td>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.</td>
</tr>
</tbody>
</table>

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)

```bash
curl -X GET -H "Authorization: Bearer ACCESS_TOKEN"
https://botplatform.example.com/bot/v1/309JF3JSIJEISIFJ0E/contactCapabilities?userContact=%2B14251234567
```

### 3.3.2.2 Responses

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td>401</td>
<td>The request is unauthorized.</td>
</tr>
<tr>
<td>404</td>
<td>The user contact or chat ID cannot be found or the given user's device is not RCS capable.</td>
</tr>
<tr>
<td>5XX</td>
<td>Server error</td>
</tr>
</tbody>
</table>

#### 3.3.2.2.1 Response body schema

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Mandatory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>capabilities</td>
<td>string array</td>
<td>No</td>
<td>The capabilities list</td>
</tr>
<tr>
<td></td>
<td>enum: [chat, fileTransfer, videoCall, geolocationPush, callComposer, chatBotCommunication]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>reason</td>
<td>Reason</td>
<td>No</td>
<td>Additional information, if any, from the Chatbot Platform.</td>
</tr>
</tbody>
</table>

#### 3.3.2.2.2 Example 1: query user capabilities

```json```
{
   "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:
3.4.1 Request URL parameters

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

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

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Mandatory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ref</td>
<td>FileData</td>
<td>Yes</td>
<td>Information about a file to be uploaded.</td>
</tr>
</tbody>
</table>

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/m4a
------WebKitFormBoundaryWfPNVh4wuWBlyEyQ
Content-Disposition: form-data; name="until"
2017-10-03T21:08:15.933Z
------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

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>202</td>
<td>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.</td>
</tr>
<tr>
<td>400</td>
<td>This is a bad request with invalid input, invalid object, etc.</td>
</tr>
<tr>
<td>401</td>
<td>The request is unauthorized.</td>
</tr>
<tr>
<td>5XX</td>
<td>Server error</td>
</tr>
</tbody>
</table>

3.4.2.2.1 Response body schema

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Mandatory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file</td>
<td>File</td>
<td>No</td>
<td>Information about the uploaded file.</td>
</tr>
<tr>
<td>reason</td>
<td>Reason</td>
<td>No</td>
<td>Additional information, if any, from the Chatbot Platform.</td>
</tr>
</tbody>
</table>

3.4.2.2.2 Example 1: upload a file response (informative)

```json
{
    "file": {
        "fileId": "MzJmajlmamVzzGZh0bmk5MHNlWmRmZTAz",
        "fileUrl": "string"
    }
}
```
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)

```bash
curl -X DELETE -H "Authorization: Bearer ACCESS_TOKEN"
https://botplatform.example.com/bot/v1/xFX3ijJ4TeiqrE3QyCMHrw/files/bot-temp-file-upload-866f331d-d1cb-489c-8154-233662b98dd5
```

3.4.3.2  Responses

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>204</td>
<td>The file has been deleted</td>
</tr>
<tr>
<td>401</td>
<td>The request is unauthorized</td>
</tr>
<tr>
<td>404</td>
<td>The file cannot be found</td>
</tr>
<tr>
<td>5XX</td>
<td>Server error</td>
</tr>
</tbody>
</table>

3.4.3.2.1  Response body schema

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Mandatory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reason</td>
<td>Reason</td>
<td>No</td>
<td>Additional information, if any, from the Chatbot Platform.</td>
</tr>
</tbody>
</table>

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)

```bash
curl -X GET -H "Authorization: Bearer ACCESS_TOKEN"
https://botplatform.example.com/bot/v1/xFX3ijJ4TeiqrE3QyCMHrw/files/bot-temp-file-upload-866f331d-d1cb-489c-8154-233662b98dd5
```
3.4.4.2 Responses

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td>401</td>
<td>The request is unauthorized</td>
</tr>
<tr>
<td>404</td>
<td>The file cannot be found</td>
</tr>
</tbody>
</table>

3.4.4.2.1 Response body schema

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Mandatory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file</td>
<td>File</td>
<td>No</td>
<td>Information about the uploaded file.</td>
</tr>
<tr>
<td>reason</td>
<td>Reason</td>
<td>No</td>
<td>Additional information, if any, from the Chatbot Platform.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Mandatory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ref</td>
<td>WebhookPayload</td>
<td>Yes</td>
<td>RCS event posted to webhook by Chatbot Platform</td>
</tr>
</tbody>
</table>

3.5.1.1.2 Example 1: receive text message (informative)

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

```json
{
    "RCSMessage": {
        "msgId": "Xs8CI3tdf",
        "textMessage": "hello world",
        "timestamp": "2017-09-26T01:33:20.315Z"
    },
    "messageContact": {
        "chatId": "93JF93SE1JF3E"
    },
    "event": "message"
}
```

3.5.1.1.4 Example 3: receive a geolocation push (informative)

```json
{
    "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)

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

```
{
    "thumbnailFileSize": 1234,
    "fileName": "f.jpg",
    "fileUrl": "http://www.example.com/files/f.jpg",
    "fileMIMEType": "image/jpeg",
    "fileSize": 1234567
}
```
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"
    }
}
```
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

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>The Callback payload is received by the Chatbot.</td>
</tr>
</tbody>
</table>
Annex B  Document Management

A.1  Document History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Brief Description of Change</th>
<th>Approval Authority</th>
<th>Editor / Company</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>22.11.2017</td>
<td>Initial release</td>
<td>TG</td>
<td>Zhinan Zhou / Samsung</td>
</tr>
</tbody>
</table>

A.2  Other Information

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document Owner</td>
<td>RCS MaaP</td>
</tr>
<tr>
<td>Editor / Company</td>
<td>Erdem Ersoz / GSMA</td>
</tr>
</tbody>
</table>

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.