Dotgo
RBM API

API DOCUMENT

Table of Contents

1. INTRODUCTION 4

1.1. OVERVIEW 4
1.2. SIGN UP 40
1.3. RBM AGENT REGISTRATION 40
1.4. AUTHENTICATION AND AUTHORIZATION 40
1.5. REFERENCES 40

2. GSMA API 41

2.1. COMMUNICATE WITH USER 41
2.1.1. SEND A MESSAGE TO THE USER 42
2.1.1.1. EXAMPLES 44
2.1.2. SEND ISTYPING INDICATION TO THE USER 53
2.1.2.1. EXAMPLE: 53
2.2. MESSAGE STATUS 54
2.2.3. QUERY STATUS OF A MESSAGE 54
2.2.3.1. EXAMPLE: 54
2.2.4. SEND 'READ' NOTIFICATION TO USERS 56
2.2.4.1. EXAMPLE 56
2.2.5. REVOKE A SENT MESSAGE 56
2.2.5.1. RICHEXAMPLE: 56
2.3. CHECK RCS CAPABILITY OF THE CONTACT/USER 57
2.3.6. EXAMPLE: 59
2.4. FILES 59
2.4.1. UPLOAD A FILE TO THE PLATFORM 59
2.4.1.1. UPLOAD A RAW FILE 61
2.4.1.1.1. EXAMPLE: 61
2.4.1.2. UPLOAD A FILE BY URL 62
2.4.1.2.1. EXAMPLE: 62
2.4.2. DELETE THE UPLOADED FILE 62
2.4.2.1. EXAMPLE: 63
2.4.3. GET STATUS OF THE UPLOADED FILE 63
2.4.3.1. EXAMPLE: 64
2.5. WEBHOOK 65
2.5.1. WEBHOOK PAYLOAD FROM DOTGO RBM PLATFORM 65
2.5.1.1. EXAMPLE 65
2.5.1.1.1. MESSAGE FROM A USER 65
2.5.1.1.2. ISTYPING MESSAGE FROM A USER 66
2.5.1.1.3. MESSAGE STATUS UPDATE 66
2.5.2. TTL EXPIRATION REVOKE SUCCESS EVENT 66
2.5.3. TTL EXPIRATION REVOKE FAILED EVENT 66
2.5.3.1.1. RESPONSE TO SUGGESTED REPLY/ACTION 67
2.5.3.1.2. RESPONSE TO SUGGESTED REPLY/ACTION INSIDE TEMPLATES 67
2.6. BULK CAPABILITY CHECK 68
2.6.1. EXAMPLE 69

3. GOOGLE APIS 71

3.1. AUTHORIZATION 71
3.2. SEND A MESSAGE TO THE USER 71
3.3. MESSAGE STATUS 78
3.3.1. SEND 'READ' NOTIFICATION TO USERS 79
3.3.2. SEND TYPING EVENT 80
3.4. REVOKE A MESSAGE 81
3.5. CHECK RCS CAPABILITY OF THE CONTACT/USER 83
3.6. UPLOAD A FILE TO THE PLATFORM 85
3.7. ADD TESTER 86
3.8. WEBHOOK 87
3.8.2.1. EXAMPLE 88
3.8.2.1.1. MESSAGE FROM A USER 88
3.8.2.1.2. ISTYPING MESSAGE FROM A USER 88
3.8.2.1.3. MESSAGE STATUS UPDATE 88
3.8.2.1.4. RESPONSE TO SUGGESTED REPLY/ACTION 89
3.8.2.1.5. RESPONSE TO SUGGESTED REPLY/ACTION OF TEMPLATES 89
3.8.2.1.6. TTL_EXPIRATION_REVOKE EVENT 90
3.8.2.1.7. TTL_EXPIRATION_REVOKE_FAILED 90

4. MEDIA TYPES AND MESSAGE LIMITS 90

4.1. MAXIMUM MESSAGE SIZE 90
4.2. MEDIA TYPE AND SIZE LIMITS WHEN SENDING A FILE 90
4.3. SIZE LIMITS FOR RICH CARD AND RICH CARD CAROUSEL 91

5. TEMPLATES 91

5.1. WHAT IS AN RCS TEMPLATE 91
5.2. TEMPLATE APPROVAL PROCESS 93

6. API THROTTLING 94

6.1. API REQUEST LIMITS EXCEEDED 95
6.1.1. GSMA API 95
6.1.2. GOOGLE API 95
6.2. RATE LIMITS FOR YOUR ACCOUNT 95

7. ERROR CODE DETAILS 96

7.1. SAMPLE ERROR MESSAGE FOR HTTP STATUS CODE 401 96
7.1.1. MAAP SPECIFIC ERRORS 96
7.2. HTTPSTATUS CODE 400 : BADREQUEST 97
7.3. HTTPSTATUS CODE 401 : UNAUTHORIZED 97
7.4. HTTPSTATUS CODE 402 : PAYMENT REQUIRED 98
7.5. HTTPSTATUS CODE 403 : FORBIDDEN 98
7.6. HTTPSTATUS CODE 404 : NOT FOUND 98
7.7. HTTPSTATUS CODE 409 : CONFLICT 99
7.8. HTTPSTATUS CODE 429 : TOO MANY REQUESTS 99

7.9. HTTPSTATUS CODE 500 : INTERNAL SERVER ERROR 99
7.10. HTTPSTATUS CODE 502 : BAD GATEWAY 99
7.11. HTTPSTATUS CODE 503 : SERVICE UNAVAILABLE 99
7.12. HTTPSTATUS CODE 504 : GATEWAY TIMEOUT 100

8. BILLING EVENTS 100

1. Introduction

1.1. Overview

Using the Dotgo RBM Platform, developers can onboard an RBM chatbot on behalf of a Brand and exchanges RCS messages with RCS enabled users using RBM APIs.

Dotgo RBM platform support RBM APIs in two industry standard API formats:

- GSMA API aligned with the [GSMA FNW.11 API] defined by GSMA, and
- Google API aligned with [Google RBM Guide and Google RBM API] as define by Google.

Further, Dotgo RBM platform supports extensions to both the GSMA API and Google API for brands to create pre-defined templates and send RCS messages to a target audience.

All Dotgo RBM APIs are REST-style HTTP APIs and receive and return JSON data. A JSON request is sent to an HTTP API endpoint.

An API call includes the below attributes

- **Method**: One of the HTTP Methods like POST, GET, DELETE, and PUT.
- **URL**: The API endpoint starting with the server_root
- **HTTP Headers**: Content-Type and authorization headers. Content-Type is application/json and sees Authorization header having access token.
- **Request Body:** JSON request data

In response to an HTTP API call, the response returned will have an HTTP Status Code such as 200, 202, 404, 500, etc. and a JSON Response.

The Chatbot needs to register it's webhook with the Dotgo RBM platform. The webhook should be secured with HTTPS. User messages, responses, and status notifications will be posted on the webhook.

Dotgo RBM API's ServerRoot:

```
ServerRoot: https://api.dotgo.com/rcs
```

1.2. Sign up

Before you get started, sign up for a Dotgo Developer account for free. Once you have submitted your Chatbot (RBM agent) details with Dotgo, you will be shared a client Id and client secret and a botID to invoke the RBM APIs via an access token.

The details are shared in the subsequent sections.

1.3. RBM Agent Registration

Step 1 To use the Dotgo RBM APIs, the brand needs to submit the agent details to Dotgo. Along with the agent details, this also includes the webhook endpoint of the brand for receiving message events and notifications from the Dotgo RBM platform.

Step 2 Once the agent is created and verified at the backend by Dotgo, the agent needs to be registered with the Dotgo RBM APIs. This will generate the necessary client ID and secret in the Dotgo SSO authentication server.

Step 3 The client Id and secret and the botID will be shared with the brand over email. The webhook URL to call back the brand will also be registered along with other details.

1.4. Authentication and Authorization

The APIs are secured by the Dotgo Auth 2 SSO service. To access GSMA APIs of the Dotgo RBM platform, an access token with Chatbot-message scope needs to be provided.

Access token can be obtained from these APIs:

```
https://auth.dotgo.com/auth/oauth/token?grant_type=client_credentials
```

To get the token, the clientId and client_secret should be sent in the Authorization header as Basic authentication (base64 encoded).

The above endpoint is rate limited in terms of the number of requests allowed per minute. By default the number of transactions per minute (TPM) will be 60 per client (A client will typically be mapped to a single bot created). In case you wish to have a higher value, please get in touch with rbm-support@dotgo.com

1.5. References

Reference	Document

GSMA FNW.11 API	GSMA FNW.11 RCS MaaP Chatbot API Specifications https://www.gsma.com/solutions-and-impact/technologies/networks/wp-content/uploads/2017/11/FNW.11_v1.0.pdf
Google RBM Guide	Google RCS Business Messaging developer docs https://developers.google.com/business-communications/rcs-business-messaging/guides
Google RBM API	Google RCS Business Messaging API https://developers.google.com/business-communications/rcs-business-messaging/reference/rest

2. GSMA API

2.1. Communicate with user

Resource endpoint: `{serverRoot}/bot/v1/{botId}/messages/async`

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

Request format:

URL: `{serverRoot}/bot/v1/OsOsQ0GwNvUdLTV9Bd/messages/async`

Authorization: Access Token obtained from Auth2 SSO service as Bearer token Obtain bearer token

Request:

```
{
   "RCSMessage":{
      "textMessage":"hello world"
   },
   "messageContact":{
      "userContact":"+914253136789"
   }
}
```

Request Parameters:

Name	Type	Description	Remarks
botId	Path variable	botId registered with the Dotgo RBM platform	

userContact	A field in the Request Body	User MSISDN in canonical form	Example: +914253136789

Response Format:

```
{
    "RCSMessage": {
        "msgId": "6cd095cd-62f6-4338-bba2-4b14b98b0537",
        "status": "pending"
    }
}
```

Response Parameters:

Name	Type	Description	Remarks
msgId	A field in the Response body	Message Identifier for the message sent	
status	A field in the response body	Status of the message	The values are as defined in the Message Status response 2.11 of [GSMA FNW.11 API].

HTTP Response Codes

Code	Description
202	The request of sending message or isTyping indication is accepted by the Platform and ready to send to the user

For error codes refer section 7

2.1.1. Send a message to the user

Various types of messages that can be sent to users, including text message, file, audio message, geolocation push, rich card, and suggested chip list are as per [GSMA FNW.11 API].

Refer to section Media Types and Message Limits for size limits and media types when sending different messages, including files, standalone rich cards and rich card carousels.

In addition to the RCS messages, any template created on the Developer portal can be sent on this endpoint. Please refer to RCS Template section for more details.

```
POST  {serverRoot}/bot/v1/OsOsQ0GwNvUdLTV9Bd/messages/async
```

This is the GSMA API. Request submitted on this endpoint will receive only Pending event and validation errors (invalid client, invalid template, rate limit etc...) as API response, any other errors (charging failure, number not found, etc...) will be sent as a failure event on the agent's callback URL.

Authorization: Access Token obtained from Auth2 SSO service as Bearer token Obtain bearer token

Request Parameters:

Name	Type	Description	Remarks
botId	Path variable	botId registered with the Dotgo RBM platform	

Message Expiry:

An Additional field available in all agent messages to support message expiry feature. To help ensure timely and relevant messages, set a message expiration. This can prevent offline users from receiving stale content when they come back online. Expiration is also a good cue to invoke your fallback messaging strategy, so users get the info they need on time.

To set a message expiration, specify one of the following fields in the agent message Json as shown in examples below:

expireTime: The exact time in UTC when the message expires. A timestamp in RFC3339 UTC "Zulu" format, with nanosecond resolution and up to nine fractional digits.

Example:

"expireTime": "2024-10-02T15:01:23Z"

ttl(time to live): The amount of time before the message expires. A duration in seconds with up to nine fractional digits, ending with 's'.

Example:

"ttl": "30s"

These fields are optional and if any older dates or invalid formats are used such messages will be rejected. Once the message expires, the platform stops trying to deliver the message, and it's automatically revoked. Based on revoke status, 'ttl_expiration_revoke' or ttl_expiration_revoke_failed event will be sent on agent webhook. Please refer to section 3.8.1.1.6 and 3.8.1.1.7 for details.

2.1.1.1. Examples

Request:

Example 1: Template Message

```
{
  "RCSMessage": {
     "templateMessage": {
        "templateCode":"template_123",
        "customParams":"{\"name\":\"user\"}"
     }
  },
  "messageContact": {
    "userContact": "+14251234567"
  },
  "ttl": "8s"
}
```

Example 2: Text Message

```
{
   "RCSMessage":{
      "textMessage":"hello world",

   },
   "messageContact":{
      "userContact":"+914253136789"
   },
   "expireTime": "2024-10-02T15:01:23Z"
}
```

Example 3: Text Message With All Suggestions

```
{
   "RCSMessage":{
      "textMessage":"Welcome to RCS",
      "suggestedChipList":{
         "suggestions":[
            {
               "reply":{
                  "displayText":"suggestion#1",
                  "postback":{
```

```
            "data":"set_by_chatbot_reply_1"
         }
      }
   },
   {
      "action":{
         "displayText":"Call",
         "postback":{
            "data":"postback_data_1234"
         },
         "dialerAction":{
            "dialPhoneNumber":{
               "phoneNumber":"+15556667777"
            }
         }
      }
   },
   {
      "action":{
         "urlAction":{
            "openUrl":{
               "url":"https://www.google.com"
            }
         },
         "displayText":"Open website or deep link",
         "postback":{
            "data":"set_by_chatbot_open_url"
         }
      }
   },
   {
      "action":{
         "calendarAction":{
            "createCalendarEvent":{
               "startTime":"2017-03-14T00:00:00Z",
               "endTime":"2017-03-14T23:59:59Z",
               "title":"Meeting",
               "description":"GSG review meeting"
            }
         },
         "displayText":"Schedule Meeting",
         "postback":{
            "data":"set_by_chatbot_create_calendar_event"
         }
      }
   },
   {
      "action":{
         "mapAction":{
            "showLocation":{
               "location":{
                  "latitude":37.4220041,
                  "longitude":-122.0862515,
                  "label":"Googleplex"
               }                     }
         },
         "displayText":"Show location on a map",
         "postback":{
            "data":"set_by_chatbot_open_map"
         }
```

```
                }
             },
             {
                "action":{
                   "mapAction":{
                      "requestLocationPush":{

                      }
                   },
                   "displayText":"Share location on a map",
                   "postback":{
                      "data":"set_by_chatbot_open_map"
                   }
                }
             }
          ]
       }
    },
    "messageContact":{
       "userContact":"+919686960276"
    }
}
```

Example 4: File Message

```
{
   "RCSMessage": {
       "fileMessage": {
       "fileUrl" : "https://konnect.kirusa.com/uploads/rcsTemplates/InstaVoice.png"
       }
  },
  "messageContact": {
      "userContact": "+14251234567"
  },
  "expireTime": "2024-10-02T15:01:23Z"
}
```

Example 5: File Message with Suggestions

```
{
    "RCSMessage": {
        "fileMessage": {
            "fileUrl" : "https://storage.googleapis.com/kitchen-sink-sample-
images/elephant.jpg"
        },
        "suggestedChipList": {
            "suggestions": [
            {
                "reply": {
                    "displayText": "suggestion#1",
```

```
            "postback": {
            "data": "set_by_chatbot_reply_1"
            }
        }
    },
    {
        "action": {
            "displayText": "Call",
            "postback": {
                "data" : "postback_data_1234"
            },
            "dialerAction": {
                "dialPhoneNumber":{
                "phoneNumber": "+15556667777"
                }
            }
        }
    },
    {
        "action": {
            "urlAction":{
                "openUrl":{
                    "url":"https://www.google.com"
                }
            },
            "displayText":"Open website or deep link",
            "postback":{
                "data":"set_by_chatbot_open_url"
            }
        }
    },
    {
        "action":{
            "calendarAction":{
                "createCalendarEvent":{
                    "startTime":"2017-03-14T00:00:00Z",
                    "endTime":"2017-03-14T23:59:59Z",
                    "title":"Meeting",
                    "description":"GSG review meeting"
                }
            },
            "displayText":"Schedule Meeting",
            "postback":{
                "data":"set_by_chatbot_create_calendar_event"
            }
```

```
                }
            },
            {
                "action":{
                    "mapAction":{
                        "showLocation":{
                            "location":{
                            "latitude":37.4220041,
                            "longitude":-122.0862515,
                            "label":"Googleplex"
                            },
                            "fallbackUrl":"https://www.google.com/maps/@37.4219162,-
122.078063,15z"
                        }
                    },
                    "displayText":"Show location on a map",
                    "postback":{
                        "data":"set_by_chatbot_open_map"
                    }
                }
            },
            {
                "action":{
                    "mapAction":{
                        "requestLocationPush":{}
                    },
                    "displayText":"Share location on a map",
                    "postback":{
                        "data":"set_by_chatbot_open_map"
                      }
                }
            }
            ]
        }
    },
    "messageContact": {
        "userContact": "+14251234567"
    }
}
```

Example 6. Rich Card with Suggested Chiplist

```
{
"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"
  }
}
```

Example 7. Rich Card Carousel with Suggested Chiplist

```
{
    "RCSMessage": {
        "richcardMessage": {
            "message": {
                "generalPurposeCardCarousel": {
                    "layout": {
                        "cardOrientation": "VERTICAL",
                        "cardWidth": "MEDIUM_WIDTH"
                    },
                    "content": [
                        {
                            "media": {
                                "mediaUrl": "https://storage.googleapis.com/kitchen-sink-sample-images/cute-dog.jpg",
                                "mediaContentType": "image/jpeg",
                                "height": "SHORT_HEIGHT",
                                "mediaFileSize": 100
                            },
                            "title": "What is RCS",
                            "description": "Rich Communication Services is messaging with more advanced features",
                            "suggestions": [
                                {
```

```
                                "action": {
                                    "displayText": "Contact",
                                    "postback": {
                                        "data": "postback_1"
                                    },
                                    "dialerAction": {
                                        "dialPhoneNumber": {
                                            "phoneNumber": "+919686960276"
                                        }
                                    }
                                }
                            },
                            {
                                "action": {
                                    "urlAction": {
                                        "openUrl": {
                                            "url": "https://www.gupshup.io"
                                        }
                                    },
                                    "displayText": "Visit Site",
                                    "postback": {
                                        "data": "set_by_chatbot_open_url"
                                    }
                                }
                            }
                        ]
                    },
                    {
                        "media": {
                            "mediaUrl": "https://storage.googleapis.com/kitchen-
sink-sample-images/cute-dog.jpg",
                            "height": "SHORT_HEIGHT",
                            "mediaFileSize": 100
                        },
                        "title": "Welcome to RCS",
                        "description": "Explore the features supported by RCS",
                        "suggestions": [
                            {
                                "reply": {
                                    "displayText": "Menu",
                                    "postback": {
                                        "data": "user_reply_1"
                                    }
                                }
                            },
```

```
                                {
                                    "action": {
                                        "displayText": "Call",
                                        "postback": {
                                            "data": "user_action_1"
                                        },
                                        "dialerAction": {
                                            "dialPhoneNumber": {
                                                "phoneNumber": "+15556667777"
                                            }
                                        }
                                    }
                                }
                            ]
                        }
                    ]
                }
            }
        },
        "suggestedChipList": {
            "suggestions": [
                {
                    "reply": {
                        "displayText": "Know more",
                        "postback": {
                            "data": "user_reply_2"
                        }
                    }
                }
            ]
        }
    },
    "messageContact": {
        "userContact": "+919686110276"
    }
}
```

Response: 202 Accepted

```
{
    "RCSMessage": {
        "msgId": "63c30986-3222-4369-a8e8-16c09b36f34f",
        "status": "pending"
    }
}
```

Response:403 Forbidden

```
{
    "RCSMessage": {
        "msgId": "937796c9-46d3-432f-b958-5d9215874f8f",
        "status": "failed",
        "timestamp": "2023-02-01T04:43:48.361Z"
    },
    "reason": {
        "text": "Invalid client id!",
        "code": 403
    }
}
```

Error Code Details:

For error codes refer section 7

2.1.2. Send isTyping indication to the user

The Chatbot can send isTyping indication to users as defined in 2.4 of [GSMA FNW.11 API].

2.1.2.1. Example:

```
POST  {serverRoot}/bot/v1/OsOsQ0GwNvUdLTV9Bd/messages/async
```

Authorization: Access Token obtained from Auth2 SSO service as Bearer token. Obtain bearer token

Request Parameters:

Name	Type	Description	Remarks
botId	Path variable	botId registered with the Dotgo RBM platform	

Request:

```
{
```

```
   "RCSMessage":{
      "isTyping":"active"
   },
   "messageContact":{
      "userContact":"+914253136789"
   }
}
```

Response:202 Accepted

```
{
    "RCSMessage": {
        "msgId": "e125527c-1af3-4158-8db9-9cc4f16d4733",
        "status": "pending"
    }
}
```

2.2. Message Status

Resource endpoint: `{serverRoot}/bot/v1/{botId}/messages/{msgId}/status`

Request format:

URL: `{serverRoot}/bot/v1/OsOsQ0GwNvUdLTV9Bd/messages/2bbedeb3-b071-473c-878b-dac4252d149b/status`

Authorization: Access Token obtained from Auth2 SSO service as Bearer token Obtain bearer token

Request Parameters:

Name	Type	Description	Remarks
botId	Path variable	botId registered with the Dotgo RBM platform	

2.2.3. Query status of a message

Although the message status is received on the 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'.

2.2.3.1. Example:

```
GET {serverRoot}/bot/v1/OsOsQ0GwNvUdLTV9Bd/messages/ddc24c2-cff5-48ac-baaa-4f286bc28061/status
```

Authorization: Access Token obtained from Auth2 SSO service as Bearer token. Obtain bearer token

Request:

No request body

Request Parameters:

Name	Type	Description	Remarks
msgId	Path variable	Reference message-id (received from the Dotgo RBM platform)	

Response: 200 OK

```
{
    "RCSMessage": {
        "msgId": "ddc24c2-cff5-48ac-baaa-4f286bc28061",
        "status": "sent",
        "timestamp": "2020-11-10T11:06:26"
    }
}
```

Response Parameters:

Name	Type	Description	Remarks
msgId	A field in the Response body	Message Identifier for the message	
status	A field in the response body	Status of the message	The values are as defined in the Message Status response 2.11 of [GSMA FNW.11 API].

HTTP Response Codes

Code	Description
200	The status of the message will be returned in the response body.

For error codes refer section 7

2.2.4. Send 'read' notification to users

A chatbot can send a 'read' notification to users for any received user message.

2.2.4.1. Example

```
PUT {serverRoot}/bot/v1/OsOsQ0GwNvUdLTV9Bd/messages/MsYMv92L4HS5GJ54rW5Xm2JA/status
```

Authorization: Access Token obtained from Auth2 SSO service as Bearer token. Obtain bearer token

Request:

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

Request Parameters:

Name	Type	Description	Remarks
msgId	Path variable	Message-Id of the received user message	

Response:204 No Content
No response body

HTTP Response Codes

Code	Description
204	The status of the message has been updated and a read notification will be sent to the user

For error codes refer section 7

2.2.5. Revoke a sent message

Sent message can be revoked only if it has not been delivered to the user. Hence the response from the platform is not the confirmation of revocation. It only indicates that the Dotgo RBM platform shall try to revoke the message.

2.2.5.1. RIchExample:

```
PUT {serverRoot}/bot/v1/OsOsQ0GwNvUdLTV9Bd/messages/MsW6S-iK49TuCRSwG=mfNP7Q/status
```

Authorization: Access Token obtained from Auth2 SSO service as Bearer token. Obtain bearer token

Request:

```
{
  "RCSMessage": {
    "status": "cancelled"
  }
}
```

Request Parameters:

Name	Type	Description	Remarks
msgId	Path variable	Reference message-id (received from the Dotgo RBM platform)	

Response:204 No Content
No response body

HTTP Response Codes

Code	Description
204	The Dotgo RBM Platform shall try to revoke the message if it has not been delivered to the user.

For error codes refer section 7

2.3. Check RCS capability of the contact/user

Resource endpoint: `{serverRoot}/bot/v1/{botId}/contactCapabilities`

This is an RCS based communication service, the Chatbot shall only communicate with the user using an 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.

Possible capabilities include 'chat', 'fileTransfer', 'videoCall', 'geolocationPush', 'callComposer' and 'chatBotCommunication'.

Request format:

URL: `{serverRoot}/bot/v1/OsQ0GwNvUdLTV9Bd/contactCapabilities`

Authorization: Access Token obtained from Auth2 SSO service as Bearer token. Obtain bearer token

Request:

No request body

Request Parameters:

Name	Type	Description	Remarks
botId	Path variable	botId registered with the Dotgo RBM platform	
userContact	Query parameter	User phone number in canonical form	Example: +914253136789 (Please note the country code prefix along with a '+')

Response Format:

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

Response Parameters:

Name	Type	Description	Remarks
capabilities	A field in the Response body	capabilities supported by contact/user	These capabilities need to be interpreted as specified in 3.3 of GSMA MaaP Chatbot API specifications.

HTTP Response Codes

Code	Description

200	Supported capabilities will be returned if the user's device is RCS capable.

For error codes refer section 7

2.3.6. Example:

```
GET {serverRoot}/bot/v1/OsQ0GwNvUdLTV9Bd/contactCapabilities? userContact=+914253136788
```

Authorization: Access Token obtained from Auth2 SSO service as Bearer token. Obtain bearer token

Request:

No request body

Response:404 NOT FOUND

```
{
    "status": "Error",
    "message": "Number is rcs disabled or Bot is not launched with number's provider "
}
```

2.4. Files

Resource endpoints: `{serverRoot}/bot/v1/{botId}/files`
`{serverRoot}/bot/v1/{botId}/files/{fileId}`

This is not for file transfer, but uploading files from Chatbot to the Dotgo RBM platform, to check the status of the uploaded file and to delete the uploaded file.

Request Parameters:

Name	Type	Description	Remarks
botId	Path variable	botId registered with the Dotgo RBM platform	

Refer to section Media Type and Size Limits when Sending a File for the type of media types and size limits supported when sending a file.

2.4.1. Upload a file to the platform

Resource endpoints: `{serverRoot}/bot/v1/{botId}/files`

There are two ways to upload files to the Dotgo RBM platform. Files can be uploaded directly to the Dotgo RBM platform or by sharing a file URL.

Request format:

URL: `{serverRoot}/bot/v1/OsQ0GwNvUdLTV9Bd/files`

Authorization: Access Token obtained from Auth2 SSO service as Bearer token. Obtain bearer token

Request: contentType - multipart/formdata

form data fields : -

'fileType'

'until '

'fileContent' or 'fileUrl'

Request Parameters:

Name	Type	Description	Remarks
fileType	A field in the form	Format of the file	Example: audio/mp4 or image/png etc..
until	A field in the form	The expiry for the file	
fileContent	A field in the form	The file that needs to be uploaded	
fileUrl	A field in the form	Valid file URL from where the file needs to be read and uploaded	

Response Format :

```
{
    "File": {
        "fileId": "GiYosOsQ0GwNvUdLTV9Bd2naXGkuz8bmDA",
        "fileUrl": "https:/stagingrcsapi.kirusa.com/OsQ0GwNvUdLTV9Bd/someFile.png",
        "fileSize": 15022,
        "status": "ready",
        "validity": "2020-11-12T11:06:26"
    }
}
```

Response Parameters:

Name	Type	Description	Remarks

fileId	A field in the Response body	The identifier for the uploaded file	
fileUrl	A field in the response body	The file URL, that can be used later as media content in the rich card or other message types such as file transfer or audio message	
status	A field in the response body	Status, if the file is ready for use	

HTTP Response Codes

Code	Description
202	The file is uploaded to the platform.

For error codes refer section 7

2.4.1.1. Upload a raw file

2.4.1.1.1. Example:

```
POST {serverRoot}/bot/v1/OsQ0GwNvUdLTV9Bd/files
```

Authorization: Access Token obtained from Auth2 SSO service as Bearer token. Obtain bearer token

Request:contentType - multipart/formdata

fileType	Image/png
until	2020-11-12T11:06:26
fileContent	picture1.png

Response: 202 Accepted

```
{
    "File": {
        "fileId": "YB5tV2prGiY2eAyIuFZ6AiEJ1zPnaXGk",
        "fileUrl": "https:/stagingrcsapi.kirusa.com/OsQ0GwNvUdLTV9Bd/picture1.png",
        "fileSize": 15022,
        "status": "ready",
        "validity": "2020-11-12T11:06:26"
```

```
    }
}
```

2.4.1.2. Upload a file by URL

2.4.1.2.1. Example:

```
POST {serverRoot}/bot/v1/OsQ0GwNvUdLTV9Bd/files
```

Authorization: Access Token obtained from Auth2 SSO service as Bearer token. Obtain bearer token

Request:contentType - multipart/formdata

fileType	Image/png
until	2020-11-12T11:06:26
fileUrl	https://somedomain.com/pictures/picture2.png

Response: 202 Accepted

```
{
    "File": {
        "fileId": "wPT9VpDqEfW98EnYDpSiuMC74yn7N7tX",
        "fileUrl":"https:/stagingrcsapi.kirusa.com/OsQ0GwNvUdLTV9Bd/picture2.png",
        "fileSize": 14052,
        "status": "ready",
        "validity": "2020-11-12T11:06:26"
    }
}
```

2.4.2. Delete the uploaded file

Resource endpoints: `{serverRoot}/bot/v1/{botId}/files/{fileId}`

A chatbot can delete the file uploaded to the platform earlier, by providing the file Id.

Request format:

URL:`{serverRoot}//bot/v1/OsQ0GwNvUdLTV9Bd/files/GiYosQ0GwNvUdLTV9Bd2naXGkuz8bmDA`

Authorization: Access Token obtained from Auth2 SSO service as Bearer token.Obtain bearer token

Request:

No request body

Request Parameters:

Name	Type	Description	Remarks
fileId	Path variable	Reference file Id (received from the Dotgo RBM platform)	

Response Format: 204 No Content

No response body

HTTP Response Codes

Code	Description
204	The file has been deleted.

For error codes refer section 7

2.4.2.1. Example:

```
DELETE {serverRoot}/bot/v1/OsQ0GwNvUdLTV9Bd/files/DqVpWEf8E7N7tnpXwPSiuMT99YDC74yn
```

Authorization: Access Token obtained from Auth2 SSO service as Bearer token.Obtain bearer token

Request:

No request body

Response: 204 No Content

2.4.3. Get status of the uploaded file

Resource endpoint: `{serverRoot}/bot/v1/{botId}/files/{fileId}`
A chatbot can retrieve the uploaded file's information. This API provides a way to check the file status along with other information. Possible file status includes 'ready' and 'expired'.

Request format:

URL: `{serverRoot}/bot/v1/OsQ0GwNvUdLTV9Bd/files/YB5tV2prGiY2eAyIuFZ6AiEJ1zPnaXGk`

Authorization: Access Token obtained from Auth2 SSO service as Bearer token.Obtain bearer token

Request:

No request body

Request Parameters:

Name	Type	Description	Remarks
fileId	Path variable	Reference file Id (received from the Dotgo RBM platform)	

Response Format:

```
{
    "File": {
        "fileId": "YB5tV2prGiY2eAyIuFZ6AiEJ1zPnaXGk",
        "fileUrl": "https:/stagingrcsapi.kirusa.com/OsQ0GwNvUdLTV9Bd/picture1.png",
        "fileSize": 15022,
        "status": "ready",
        "validity": "2020-11-12T11:06:26"
    }
}
```

Response Parameters:

Name	Type	Description	Remarks
fileId	A field in the Response body	The identifier for the uploaded file	
fileUrl	A field in the response body	The file URL, that can be used later as media content in the rich card or other message types such as file transfer or audio message	
status	A field in the response body	Status, if the file is ready for use	Values: ready, expired

HTTP Response Codes

Code	Description
200	File information will be returned

For error codes refer section 7

2.4.3.1. Example:

GET{serverRoot}/bot/v1/OsQ0GwNvUdLTV9Bd/files/YB5tV2prGiY2eAyIuFZ6AiEJ1zPnaXGk

Authorization: Access Token obtained from Auth2 SSO service as Bearer token.Obtain bearer token

Request:

No request body

Response:200 OK

```
{
    "File": {
        "fileId": "YB5tV2prGiY2eAyIuFZ6AiEJ1zPnaXGk",
        "fileUrl": "https:/stagingrcsapi.kirusa.com/OsQ0GwNvUdLTV9Bd/picture1.png",
        "fileSize": 15022,
        "status": "expired",
        "validity": "2020-11-12T11:06:26"
    }
}
```

2.5. Webhook

Webhook is the callback API provided by the Chatbot. Dotgo RBM Platform uses the webhook exposed by Chatbot to send an HTTP POST payload when certain RCS events occur.

A chatbot may receive the following events from the Dotgo RBM platform:

1. message from a user (text, file, location, or audio message)
2. isTyping notification from a user
3. message status update
4. response to suggested reply or action

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

2.5.1. Webhook payload from Dotgo RBM Platform

The payload that can be sent from the Dotgo RBM platform to Chatbot's webhook is as per the WebhookPayload model specified in 2.15 of [GSMA FNW.11 API].

2.5.1.1. Example

2.5.1.1.1. Message from a user

```
{
   "RCSMessage": {
      "msgId":"MsWHg8WU3cTjK7scyQ1bv-aA",
      "textMessage":"hello world",
      "timestamp":"2020-11-12T09:16:29.527061Z"
   },
```

```
   "messageContact": {
      "userContact":"+914253136789"
   },
   "event":"message"
}
```

2.5.1.1.2. isTyping message from a user

```
{
   "RCSMessage":{
      "isTyping":"active",
      "timestamp":"2020-11-12T09:34:44.876908Z"
   },
   "messageContact":{
      "userContact":"+914253136789"
   },
   "event":"isTyping"
}
```

2.5.1.1.3. Message status update

```
{
   "RCSMessage":{
      "msgId":"334a5db9-b19d-46c4-80e1-ff662ba2a982",
      "status":"delivered",
      "timestamp":"2020-11-12T09:55:49.444158Z"
   },
   "messageContact":{
      "userContact":"+914253136789"
   },
   "event":"messageStatus"
}
```

2.5.2. ttl expiration revoke success event

```
{
   "RCSMessage":{
      "msgId":"334a5db9-b19d-46c4-80e1-ff662ba2a982",
      "status":"ttl_expiration_revoked",
      "timestamp":"2020-11-12T09:55:49.444158Z"
   },
   "messageContact":{
      "userContact":"+914253136789"
   },
   "event":"messageStatus"
}
```

2.5.3. ttl expiration revoke failed event

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

```
      "msgId":"334a5db9-b19d-46c4-80e1-ff662ba2a982",
      "status":"ttl_expiration_revoke_failed”,
      "timestamp":"2020-11-12T09:55:49.444158Z"
   },
   "messageContact":{
      "userContact":"+914253136789"
   },
   "event":"messageStatus"
}
```

2.5.3.1.1. Response to suggested reply/action

```
{
   "RCSMessage":{
      "msgId":"Ms4xAa25HvQfGpX8dWXdNXDw",
      "timestamp":"2020-11-12T09:34:38.553753Z",
      "suggestedResponse":{
         "response":{
            "reply":{
               "displayText":"Visit a website",
               "postback":{
                  "data":"set_by_chatbot_reply_yes"
               }
            }
         }
      }
   },
   "messageContact":{
      "userContact":"+914253136789"
   },
   "event":"response"
}
```

2.5.3.1.2. Response to suggested reply/action inside templates

Template description has been provided in Template APIs.

```
{
   "RCSMessage":{
      "msgId":"MxNsfgtg86T-6-=o2i7P3pGw",
      "timestamp":"2023-06-20T15:39:30.475341Z",
      "suggestedResponse":{
         "response":{
            "action":{
               "displayText":"Reach Us",
               "postback":{
```

```
                "data":"user_clicked_Reach_Us",

"metadata":"{\"suggestionType\":\"url_action\",\"templateType\":\"carousel\",\"cardIndex
\":1,\"suggestionIndex\":2,\"msgId\":\"1ed15c61-7fd1-436a-a127-
0baa8d91cdb2\",\"a2pMsgDate\":\"2023-07-03T15:22:00.356\",\"templateCode\":\"test
template\"}"
              }
            }
          }
        }
    },
    "messageContact":{
       "userContact":"+919976878767"
    },
    "event":"response"
}
```


The metadata fields will be additionally sent as part of the payload to the agent webhook in case the user responses are within A2P message templates. This will include the following:

SuggestionType:	reply, dialer_action, url_action
templateType:	text_message, rich_card, carousel
cardIndex:	in case of a carousel, the index of the card for which the user event is generated (index starts from 0)
suggestionIndex:	index of the suggestion starting from 0
msgId:	the message Id of the original A2P message
a2pMsgDate:	date in which the original A2P message was sent
templateCode:	name of the template

2.6. Bulk Capability Check

This is the API which returns RBM enabled users from the given list of users.

To get the number of RBM-reachable users, do a bulk capability check. Bulk checks include whether a phone number is reachable but not which capabilities or features a phone number supports (Please refer to the single capability check API in Section 2.3 to understand what is meant by capabilities). You can specify up to 10,000 phone numbers per bulk capability check. To check more numbers, perform multiple checks.

Bulk capability checks respond with the specified numbers that are reachable by the agent.

Bulk capability checks may fail for several reasons, such as the underlying MAAP not supporting it, multiple MAAPs being configured, or the number of requests exceeding the allowed limit.

Note: This is a common API for both Google and GSMA and is applicable for only **launched bots.**

2.6.1. Example

```
POST {serverRoot}/bot/v1/OsOsQ0GwNvUdLTV9Bd/rcsEnabledContacts
```

Authorization: Access Token obtained from Auth2 SSO service as Bearer token.

Request:

```
{
  "users": [
    "+919687895543",
    "+919686960876",
    "+919688757768"
  ]
}
```

Request Parameters:

Name	Type	Description	Remarks
botId	Path variable	botId registered with the Dotgo RBM platform	

Response:

Example 1. 200 OK

```
{
  "rcsEnabledContacts": [
    "+919686960876"
  ]
}
```

Example 2. 400 Bad Request

```
{
  "Error": "Requested users shouldn't be more than 10000 in a request"
}
```

Example 3: 424 Failed Dependency

```
{
   "status":"Error",
   "message":"bulk capability check is not supported for agent with multiple MaaPs"
}
```

Example 4: 424 Failed Dependency

```
{
   "status":"Error",
   "message":"telcel MaaP doesn't support bulk capability check"
}
```

HTTP Response Codes

Code	Description

200	RCS enabled users from the provided list of users

For error codes refer section 7

3. Google APIs

Dotgo RBM Platform also provides Google APIs as per Google RBM API specifications for the ease of use for developers and brands who may have already integrated with Google RBM APIs for testing, and now wants to integrate with the Dotgo RBM Platform to launch with multiple carriers in different countries. for greater reach. With these APIs, they can seamlessly integrate with Dotgo's Google API (even for carriers that have RCS from anther vendor).

3.1. Authorization

To access Google APIs of the Dotgo RBM platform, an access token with 'Chatbot-message' and 'google' scope needs to be provided.

Register a ServiceClient with scope 'Chatbot-message' and 'google' to receive OAuth2client_credentials. Using the credentials obtained from SSO, an access token can be generated which in turn should be used as an Authorization Bearer token, to access Google API of RCS platform.

Access token can be obtained from these APIs :

```
https://auth.dotgo.com/auth/oauth/token?grant_type=client_credentials
```

To get the token, the clientId and client_secret should be sent in the Authorization header as Basic authentication (base64 encoded).

3.2. Send a Message to the user

This is the API used to send messages to users. Various types of messages can be sent to users, including text message, file, rich card, and carousel as per Google API specifications. Suggestions can be sent along with any of the mentioned message types.

Refer to section Media Types and Message Limits for size limits and media types when sending different messages, including files, standalone rich cards and rich card carousels.

In addition to the RCS messages, any template created on the Developer portal can be sent on this endpoint. Please refer to Rich Template section for more details.

```
POST{serverRoot}/v1/phones/{phone_number}/agentMessages/async?botId={bot_id}&messageId=
{message_id}
```

This is the Google API. Request submitted on this endpoint will receive only Pending event and validation errors (invalid client, invalid template, rate limit etc...) as API response, any other errors (charging failure, number not found, etc...) will be sent as a failure event on the agent's callback URL.

Request Parameters:

Name	Type	Description	Remarks
phone_number	Path variable	User MSISDN in canonical form	Example: +914253136789
botId	Query parameter	botId registered with the Dotgo RBM platform	
messageId	Query parameter	Unique id for the message (not mandatory)	Optional. In case of duplicate messageId, request will be rejected. If messageId is not provided, Platform will assign one and include it in response.

Message Expiry:

An Additional field available in all agent messages to support message expiry feature. To help ensure timely and relevant messages, set a message expiration. This can prevent offline users from receiving stale content when they come back online. Expiration is also a good cue to invoke your fallback messaging strategy, so users get the info they need on time.

To set a message expiration, specify one of the following fields in the agent message JSON:

expireTime: The exact time in UTC when the message expires. A timestamp in RFC3339 UTC "Zulu" format, with nanosecond resolution and up to nine fractional digits.

Example:

"expireTime": "2024-10-02T15:01:23Z"

ttl(time to live): The amount of time before the message expires. A duration in seconds with up to nine fractional digits, ending with 's'.

Example:

"ttl": "30s"

These fields are optional and if any older dates or invalid formats are used such messages will be rejected. Once the message expires, the platform stops trying to deliver the message, and it's

automatically revoked. Based on revoke status, 'ttl_expiration_revoke' or ttl_expiration_revoke_failed event will be sent on agent webhook. Please refer to section 3.8.1.1.6 and 3.8.1.1.7 for details.

Request format:

URL:
`POST https://api.dotgo.com/rcs/v1/phones/+914253136789/agentMessages/async?botId=OsOsQ0GwNvUdLTV9Bd&messageId=d2f81052-37c7-469c-997b-9576023b4220`

Authorization: Access Token obtained from Auth2 SSO service as Bearer token.Obtain bearer token

Request:

Example 1: Template Message

```
{
   "contentMessage":{
      "templateMessage":{
          "templateCode":"template_123",
          "customParams":"{\"name\":\"user\"}"
      }
   },
   "ttl": "10s"
}
```

Example 2: Text

```
{
  "contentMessage" : {
    "text" : "Welcome to RCS chat!"
  },
  "expireTime": "2024-10-02T15:01:23Z"
}
```

Example 3: Text with Suggestions

```
{
    "contentMessage": {
        "text": "Welcome to RCS chat!",
        "suggestions": [
            {
                "reply": {
                    "text": "what is RCS?",
                    "postbackData": "user_reply_what_is_rcs"
                }
            },
            {
                "action": {
                    "text": "visit our website",
                    "postbackData": "user_action_open_url",
                    "openUrlAction": {
                        "url": "https://www.dotgo.com"
                    }
                }
            },
```

```
            {
                "action": {
                    "text": "Show location on a map",
                    "postbackData": "set_by_chatbot_open_map",
                    "viewLocationAction": {
                        "latLong": {
                            "latitude": 37.4220041,
                            "longitude": -122.0862515
                        },
                        "label": "Googleplex"
                    }
                }
            },
            {
                "action": {
                    "text": "Share location on a map",
                    "postbackData": "set_by_chatbot_open_map",
                    "shareLocationAction": {}
                }
            }
        ]
    }
}
```

Example 4: File with URL

```
{
  "contentMessage" : {
    "contentInfo" : {
      "fileUrl" : "https://konnect.kirusa.com/uploads/rcsTemplates/InstaVoice.png"
    }
  }
}
```

Example 5: File with Suggestions

```
{
  "contentMessage" : {
    "contentInfo" : {
      "fileUrl" : "https://konnect.kirusa.com/uploads/rcsTemplates/InstaVoice.png"
    },
    "suggestions" : [ {
      "reply" : {
        "text" : "what is RCS?",
        "postbackData" : "user_reply_what_is_rcs"
      }
    },
    {
      "action" : {
        "text" : "visit our website",
        "postbackData" : "user_action_open_url",
        "openUrlAction" : {
          "url" : "https://www.dotgo.com"
        }
      }
    }]
  },
```

```
  "expireTime": "2014-10-02T15:01:23Z"
}
```

Example 6: File with ID (uploaded file)

```
{
  "contentMessage" : {
    "fileName":"VLXtA7s35cGmyq6g9TcqCSEn2Uqi9QLR"
  }
}
```

Example 7: Rich Card

```
{
  "contentMessage" : {
    "richCard" : {
      "standaloneCard" : {
        "cardContent" : {
          "media" : {
            "contentInfo" : {
              "fileUrl" :
"https://stagingchannels.kirusa.com/vobolo/blogs/9827/11914487_1609907755880.mp4",
              "forceRefresh" : false,
              "thumbnailUrl" :
"https://stagingchannels.kirusa.com/vobolo/blogs/9827/11914487_1609907755880_preview_500
x500.png"
            },
            "height" : "MEDIUM"
          },
          "title" : "Celebrity",
          "suggestions" : [ {
            "reply" : {
              "text" : "Like",
              "postbackData" : "user_like"
            }
          } ]
        },
        "thumbnailImageAlignment" : "LEFT",
        "cardOrientation" : "VERTICAL"
      }
    },
    "suggestions" : [ {
      "reply" : {
        "text" : "Know more",
        "postbackData" : "user_query"
      }
    } ]
  }
}
```

Example 8: Rich Card Carousel

```
{
    "contentMessage": {
        "richCard": {
            "carouselCard": {
                "cardWidth": "MEDIUM",
                "cardContents": [
                    {
                        "title": "This is the first rich card in a carousel.",
                        "description": "This is the description of the rich card.",
                        "media": {
```

```
                            "height": "SHORT",
                            "contentInfo": {
                                "fileUrl": "https://storage.googleapis.com/kitchen-sink-
sample-images/elephant.jpg",
                                "forceRefresh": false
                            }
                        },
                        "suggestions": [
                            {
                                "reply": {
                                    "text": "what is RCS?",
                                    "postbackData": "user_reply_what_is_rcs"
                                }
                            },
                            {
                                "action": {
                                    "text": "visit our website",
                                    "postbackData": "user_action_open_url",
                                    "openUrlAction": {
                                        "url": "https://www.dotgo.com"
                                    }
                                }
                            }
                        ]
                    },
                    {
                        "title": "This is the second rich card in a carousel.",
                        "media": {
                            "height": "SHORT",
                            "contentInfo": {
                                "fileUrl": "https://storage.googleapis.com/kitchen-sink-
sample-images/elephant.jpg",
                                "forceRefresh": false
                            }
                        },
                        "suggestions": [
                            {
                                "action": {
                                    "text": "Show location on a map",
                                    "postbackData": "set_by_chatbot_open_map",
                                    "viewLocationAction": {
                                        "latLong": {
                                            "latitude": 37.4220041,
                                            "longitude": -122.0862515
                                        },
                                        "label": "Googleplex"
                                    }
                                }
                            },
                            {
                                "action": {
                                    "text": "Share location on a map",
                                    "postbackData": "set_by_chatbot_open_map",
                                    "shareLocationAction": {}
                                }
                            }
                        ]
                    }
                ]
            }
```

```
        },
        "suggestions": [
            {
                "reply": {
                    "text": "what is RCS?",
                    "postbackData": "user_reply_what_is_rcs"
                }
            }
        ]
    }
}
```

Response:
Example 1:
200 OK

```
{
    "name": "phones/+919686960276/agentMessages/19e538c9-69ad-4c76-ac20-3e092a87f97a",
    "sendTime": "2023-01-16T07:37:29.787Z",
    "contentMessage": {
        "text": "Welcome to RCS chat!"
    }
}
```

Example 2:

```
{
    "senderPhoneNumber": "+919686960276",
    "eventType": "FAILED",
    "reason": "Invalid client id!",
    "sendTime": "2023-02-01T04:45:44.206Z",
    "messageId": "00b53bde-9760-41f5-8f11-7a2f25b6b2ac",
    "code": 403
}
```

Error Code Details:

For error codes refer section 7

Response Parameters:

Name	Type	Description	Remarks
messageId	A field in the Response body	Message Identifier for the message sent	

eventType	A field in the response body	Status of the message	The values are as defined in Google user event types. Additional to that there could be "PENDING" and "FAILED" events.
senderPhoneNumber	A field in the response body	User MSISDN in canonical form	Example: +914253136789
eventId	A field in the response body	Unique Identifier for the event	
message	A field in the response body	Error message if request fails	

HTTP Response Codes

Code	Description
200	The request of sending message or isTyping indication is accepted by the Platform and ready to send to the user

For error codes refer section 7

3.3. Message Status

Agent generated events for user messages can be sent through this API. Agents can send read event for a user message or typing indication to users.

Resource endpoint:
`{serverRoot}/rcs/v1/phones/{phone_number}/agentEvents?botId={bot_id}&eventId={event_id}`

Authorization: Access Token obtained from Auth2 SSO service as Bearer token.Obtain bearer token

Request Parameters:

Name	Type	Description	Remarks
phone_number	Path variable	User MSISDN in canonical form	Example: +914253136789
botId	Query parameter	botId registered with the Dotgo RBM platform	
eventId	Query parameter	Unique identifier for event	Optional.

3.3.1. Send 'read' notification to users

A chatbot can send a 'read' notification to the end user for any received user message. This event indicates that a message has been opened or acknowledged. To users, this event appears as a read receipt for a specific message.

Request format:
URL:
```
POST
https://api.dotgo.com/rcs/v1/phones/+914253136789/agentEvents?botId=OsOsQ0GwNvUdLTV9Bd&eventId=6924cc84-dc0b-4679-a4ab-50c0875b309f
```

Authorization: Access Token obtained from Auth2 SSO service as Bearer token.Obtain bearer token

Request:
```
{
  "eventType": "READ",
  "messageId": "6924cc84-dc0b-4679-a4ab-50c0875b309f "
}
```
Response: 200 OK
No response body

HTTP Response Codes

Code	Description
200	The status of the message has been updated and a read notification will be sent to the user

For error codes refer section 7

3.3.2. Send Typing Event

The Chatbot can send typing indication to users. To a user, this event appears as a typing indicator and lets him know that agent is composing a message. The typing indicator expires after a short time (approximately 20 seconds) or when the user's device receives a new message from your agent. Your agent can send multiple IS_TYPING events to reset the typing indicator's expiration timer.

Request format:

URL:

```
POST
https://api.dotgo.com/rcs/v1/phones/+914253136789/agentEvents?botId=OsOsQ0GwNvUdLTV9Bd&eventId=6924cc84-dc0b-4679-a4ab-50c0875b309f
```

Authorization: Access Token obtained from Auth2 SSO service as Bearer token.Obtain bearer token

Request:

```
{
  "eventType" : "IS_TYPING"
}
```

Response: 200 OK

No response body

HTTP Response Codes

Code	Description
200	The isTyping event has been sent to user

For error codes refer section 7

3.4. Revoke a Message

Agent can revoke messages that it has sent but that have not been delivered to the user. There is a small chance that your bot may initiate a revocation while the MaaP is in the process of delivering the message. In these rare cases, your bot receives a DELIVERED event for the message shortly after initiating the revocation request.

Resource endpoint:
`{serverRoot}/rcs/v1/phones/{phone_number}/agentMessages/{message_id}?botId={bot_id}`

Request parameter:

Name	Type	Description	Remarks
botId	Query parameter	botId registered with the Dotgo RBM platform	
phone_number	Path variable	User msisdn in canonical form	Example: +914253136789
message_id	Path variable	Identifier for the message to be revoked.	

Request format:

URL:
```
DELETE
https://api.dotgo.com/rcs/v1/phones/+914253136789/agentMessages/{message_id}?botId=
OsOsQ0GwNvUdLTV9Bd
```

Authorization: Access Token obtained from Auth2 SSO service as Bearer token.Obtain bearer token

Request: request body is optional. Include only if conversationId is available.
```
{
  "conversationId" : "a5a96c284d90ff352ac415e5a8ec213a"
}
```

Response Parameters:

Name	Type	Description	Remarks

conversationId	A field in the Response body	Identifier for the conversation. If request is sent through Vodafone MaaP, platform includes a conversationId in response. Agent must record this and send it back during revoke.	Optional. Include only if conversationId is available to agent(in case of Vodafone MaaP it will be available.)

Response:

Example 1:
200 OK
No response body

Example 2: if messageId Not found.
404 NOT FOUND

```
{
    "status": "Error",
    "message": "Not found : Message Id"
}
```

Response Parameters:

Name	Type	Description	Remarks
Status	A field in the Response body	Status of the request	
Message	A field in the Response body	Reason for failure	

HTTP Response Codes

Code	Description
200	Revoke request has been accepted and platform will try to revoke if it's not delivered to user yet.

For error codes refer section 7

3.5. Check RCS capability of the contact/user

In RCS based communication service, the Chatbot shall only communicate with the user using an RCS capable device. The Chatbot can conduct the RCS capability discovery to learn about whether the given user's device is RCS capable or not.

Possible capabilities include:
"REVOCATION",
"RICHCARD_STANDALONE",
"RICHCARD_CAROUSEL",
"ACTION_CREATE_CALENDAR_EVENT",
"ACTION_DIAL",
"ACTION_OPEN_URL",
"ACTION_SHARE_LOCATION",
"ACTION_VIEW_LOCATION".

Resource endpoint:
`{serverRoot}/rcs/v1/phones/{phone_number}/capabilities?botId={bot_id}&requestId={request_id}`

Request Parameters:

Name	Type	Description	Remarks
botId	Query parameter	botId registered with the Dotgo RBM platform	
phone_number	Path variable	User msisdn in canonical form	Example: +914253136789
requestId	Query parameter	Unique identifier for request	Optional.

Request format:

URL:
```
GET
https://api.dotgo.com/rcs/v1/phones/+914253136789/capabilities?botId=
OsOsQ0GwNvUdLTV9Bd&requestId= 2918644c-9b58-4cf9-a598-3eec750ad0e3
```

Authorization: Access Token obtained from Auth2 SSO service as Bearer token.Obtain bearer token

Request:

No request body

Response:

Example 1: 200 - Device capabilities will be returned

```
{
  "features": [
    "REVOCATION",
    "RICHCARD_STANDALONE",
    "RICHCARD_CAROUSEL",
    "ACTION_CREATE_CALENDAR_EVENT",
    "ACTION_DIAL",
    "ACTION_OPEN_URL",
    "ACTION_SHARE_LOCATION",
    "ACTION_VIEW_LOCATION",
  ]
}
```

Exampe 2: 404 - if RCS disabled, No capabilities.

```
{
    "error": {
        "code": 404,
        "message": "Requested entity was not found.",
        "status": "NOT_FOUND"
    }
}
```

Response Parameters:

Name	Type	Description	Remarks
features	A field in the Response body	capabilities supported by contact/user	These capabilities need to be interpreted as specified in Google API specifications.

HTTP Response Codes

Code	Description
200	User is RCS enabled and supported capabilities will be returned

For error codes refer section 7

3.6. Upload a file to the platform

When agent sends a message with an image or video, agent must provide a publicly accessible URL for the content. If not, agent can upload a media file binary to Dotgo CDN and then send the media in a message. The Dotgo RCS platform keeps files for 60 days, and the API returns a file Id that your agent can include in messages to users. After 60 days, the file will be removed from CDN.

Refer to section Media Type and Size Limits when Sending a File for the type of media types and size limits supported when sending a file.

Resource endpoint:
`{serverRoot}/rcs/upload/v1/files?botId={botId}`

Name	Type	Description	Remarks
botId	Query parameter	botId registered with the Dotgo RBM platform	

Request format:

URL:
`POST https://api.dotgo.com/rcs/upload/v1/files?botId={botId}`

Authorization: Access Token obtained from Auth2 SSO service as Bearer token.Obtain bearer token

Request:
binary
contentType – content type of the file (image/jpeg, video/mp4, image/png, etc..)
select the file

Response:

Example 1: returns Identifier for the uploaded file.
200 OK

```
{
  "name": "9mEFZnVTBsycGwtVTH7BovQwuz0Msr9p"
}
```

Example 2: Failed to save the file.

500 INTERNAL SERVER ERROR

```
{
    "status": "Error",
    "message": "Processing failed : Unable to store the file."
}
```

Response Parameters:

Name	Type	Description	Remarks
name	A field in the Response body	Unique identifier for the uploaded file	This value can be used in file messages, instead or file url.
status	A field in the Response body	Status of the request	
message	A field in the Response body	Reason for failure	

HTTP Response Codes

Code	Description
200	Revoke request has been accepted and platform will try to revoke if it's not delivered to user yet.

For error codes refer section 7

3.7. Add Tester

Chatbots can use this feature to register a tester device.

Resource endpoint:
`{serverRoot}/rcs/v1/phones/{phone_number}/testers?botId={bot_id}`

Request Parameters:

Name	Type	Description	Remarks
botId	Query parameter	botId registered with the Dotgo RBM platform	

phone_number	Path variable	User msisdn in canonical form	Example: +914253136789

Request format:

URL:

```
POST https://api.dotgo.com/rcs/v1/phones/{phone_number}/testers?botId=OsOsQ0GwNvUdLTV9Bd
```

Authorization: Access Token obtained from Auth2 SSO service as Bearer token.Obtain bearer token

Request:

No request body

Response :

```
{
    "statusCode": "Success",
    "response": "Submitted, Tester Invite Sent"
}
```

HTTP Response Codes

Code	Description
200	If tester added, success message in the response body. If failed to add tester, a failure message along with the reason will be returned.

For error codes refer section 7

3.8. Webhook

Webhook is the callback API provided by the Chatbot. Dotgo RBM Platform uses the webhook exposed by Chatbot to send an HTTP POST payload when certain RCS events occur.

A chatbot may receive the following events from the Dotgo RBMplatform:

1. message from a user (text, file, location, or audio message)
2. isTyping notification from a user
3. message status update
4. response to suggested reply or action

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

The payload that can be sent from the Dotgo RBM platform to the developer Chatbot's webhook is as defined in Google API specifications.

Every request sent on webhook will be base64 encoded in case of google format and set to 'data' field as below.


```
{
       "subscription": "projects/rbm-stagingtestagent-ngu7dtz/subscriptions/rbm-agent-
       subscription",
       "message": {
         "data":
       "ewogICJzZW5kZXJQaG9uZU51bWJlciI6ICIrOTE5Njg2OTYwMjc2IiwKICAibWVzc2FnZUlkIjogIk14SWdU
       LTRwV0NRdTJ0SDloTWtTVGxqUSIsCiAgInNlbmRUaW1lIjogIjIwMjItMTEtMTdUMDU6MTU6MTQuMzg0OTYzW
       iIsCiAgInRleHQiOiAiVGVzdCIsCiAgImFnZW50SWQiOiAic3RhZ2luZ3Rlc3RhZ2VudEByYm0uZ29vZyIKfQ
       ==",
         "attributes": {
             "project_number": "129459109761",
             "product": "RBM",
             "message_type": "TEXT",
             "business_id": "stagingtestagent@rbm.goog",
             "type": "message"
         },
         "messageId": "6267774379637729",
         "publishTime": "2022-11-17T05:15:14.538Z"
         }
}
```


3.8.2.1. Example

3.8.2.1.1. Message from a user


```
{
  "senderPhoneNumber": "PHONE_NUMBER",
  "messageId": "0a99d150-aae7-4247-aa07-a92cdaaf8ed3",
  "sendTime": "2018-12-31T15:01:23.045123456Z",
  "text": "Hello, world!",
  "agentId": " stagingtestagent@rbm.goog"
}
```


3.8.2.1.2. isTyping message from a user


```
{
  "senderPhoneNumber": "+914253136789",
  "eventType": "IS_TYPING",
  "eventId": "47ace754-3191-4101-b494-658cfb314881",
  "sendTime": "2018-12-31T15:01:23.045123456Z"
  "agentId": " stagingtestagent@rbm.goog"
}
```


3.8.2.1.3. Message status update


```
{
  "senderPhoneNumber": "+914253136789",
```

```
  "eventType": "DELIVERED",
  "eventId": "fa2fe5a2-d9a9-4d83-87d3-302ae1014610",
  "messageId": "57bed79e-55ba-46fe-b88a-2755aaee77fc",
  "agentId": " stagingtestagent@rbm.goog"
}
```

3.8.2.1.4. Response to suggested reply/action

```
{
  "senderPhoneNumber": "+914253136789",
  "messageId": "4a1d7c74-2b3c-4ec7-b6be-ed7205a15aa3",
  "sendTime": "2018-12-31T15:01:23.045123456Z",
  "agentId": " stagingtestagent@rbm.goog",
  "suggestionResponse": {
    "postbackData": "suggestion_1",
    "text": "Suggestion #1" }
}
```

3.8.2.1.5. Response to suggested reply/action of templates

Template description has been provided in the next section.

```
{
   "suggestionResponse":{
      "postbackData":"user_clicked_Reach_Us",
      "text":"Reach Us",
      "type":"ACTION",

"metadata":"{\"suggestionType\":\"url_action\",\"templateType\":\"carousel\",\"cardIndex
\":1,\"suggestionIndex\":2,\"msgId\":\"1ed15c61-7fd1-436a-a127-
0baa8d91cdb2\",\"a2pMsgDate\":\"2023-07-03T15:22:00.356\",\"templateCode\":\"test
template\"}"
   },
   "senderPhoneNumber":"+919986473361",
   "messageId":"MxNsfgtg86T-6-=o2i7P3pGw",
   "sendTime":"2023-06-20T15:39:30.475341Z",
   "agentId":"dotgo-gupshup_r0mhjgkx_agent@rbm.goog"
}
```

The metadata fields will be additionally sent as part of the payload to the agent webhook in case the user responses are within A2P message templates. This will include the following:

SuggestionType:	reply, dialer_action, url_action
templateType:	text_message, rich_card, carousel
cardIndex:	in case of a carousel, the index of the card for which the user event is generated (index starts from 0)
suggestionIndex:	index of the suggestion starting from 0
msgId:	the message Id of the original A2P message
a2pMsgDate:	date on which the original A2P message was sent
templateCode:	name of the template

3.8.2.1.6. ttl_expiration_revoke event

The message has expired and was successfully revoked. In this case the money will be refunded for such messages.

```
{
  "phoneNumber": "+919986473361" ,
  "messageId": ”a26d6809-9e9a-4c77-ad82-e55b35b80ef1”,
  "agentId": "dotgo-gupshup_r0mhjgkx_agent@rbm.goog",
  "eventType": "TTL_EXPIRATION_REVOKED",
  "eventId": "MxNsfgtg86T-6-=o2i7tryGw",
  "sendTime": "2023-06-20T15:39:30.475341Z"
}
```

3.8.2.1.7. ttl_expiration_revoke_failed

The message has expired, but it was not revoked.)

```
{
  "phoneNumber": "+919986473361",
  "messageId": ”b6586809-9e9a-4c77-ad82-e55b35b80ef1”,
  "agentId": "dotgo-gupshup_r0mhjgkx_agent@rbm.goog",
  "eventType": "TTL_EXPIRATION_REVOKE_FAILED",
  "eventId": "MxNsfgtg8sjfjk6=o2i7P3pGw",
  "sendTime": "2023-06-20T15:39:30.475341Z"
}
```

4. Media Types and Message Limits

This section provides guidance of the maximum size of various messages. This is common for both the GSMA API and the Google API.

4.1. Maximum Message Size

The maximum size of text that can be sent in an RBM message is 256KB. This includes the text that makes up the JSON payload when sending structured messages.

4.2. Media Type and Size Limits when Sending a File

The following media types are supported when sending a file. The max size mentioned below apply for sending direct file messages (and not for rich cards/carousels).

Media Type	Document Type	Extension	Max Size	Works with rich cards
application/pdf	PDF	.pdf	100 MB	No
image/jpeg	JPEG	.jpeg, .jpg	100 MB	Yes
image/gif	GIF	.gif	100 MB	Yes
image/png	PNG	.png	100 MB	Yes
video/h263	H263 video	.h263	100 MB	Yes
video/m4v	M4V video	.m4v	100 MB	Yes
video/mp4	MP4 video	.mp4	100 MB	Yes
video/mpeg4	MPEG-4 video	.mp4, .m4p	100 MB	Yes
video/mpeg	MPEG video	.mpeg	100 MB	Yes

video/webm	WEBM video	.webm	100 MB	Yes

4.3. Size Limits for Rich Card and Rich Card Carousel

The recommended and maximum size limits apply for standalone rich cards and carousels:

Type	Size
Recommended maximum size of image in a Standalone Rich Card	2 MB
Recommended maximum size of video in a Standalone Rich Card	10 MB
Recommended optimal size of thumbnail in a Standalone Rich Card	40 KB
Recommended maximum size of thumbnail in a Standalone Rich Card	100 KB
Recommended maximum size of image in a Rich Card Carousel	1 MB
Recommended maximum size of video in a Rich Card Carousel	5 MB
Recommended optimal size of thumbnail in a Rich Card Carousel	40 KB
Recommended maximum size of thumbnail in a Rich Card Carousel	100 KB

5. Templates

Dotgo RBM platform lets you add rich media such as images, carousels, and videos, along with customized branding, action buttons, and suggested replies into the notifications and promotional messages.

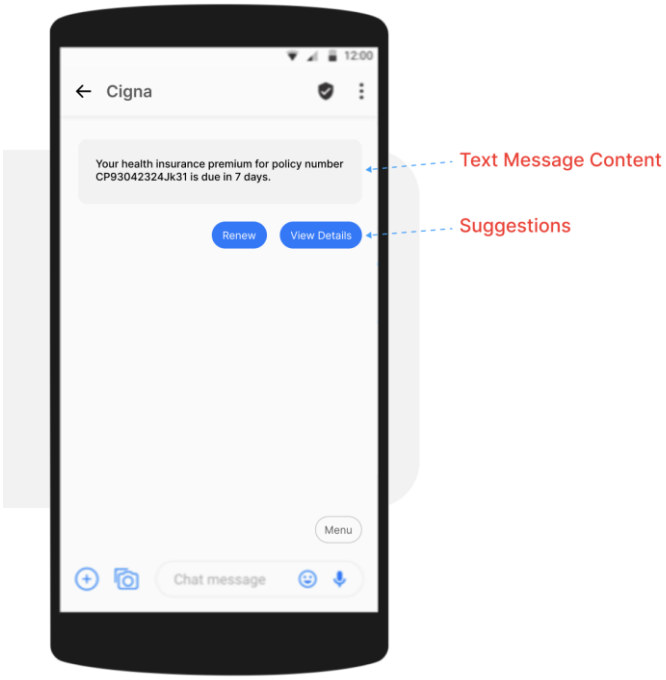
These template messages are delivered using the underlying RBM APIs.

5.1. What is an RCS Template

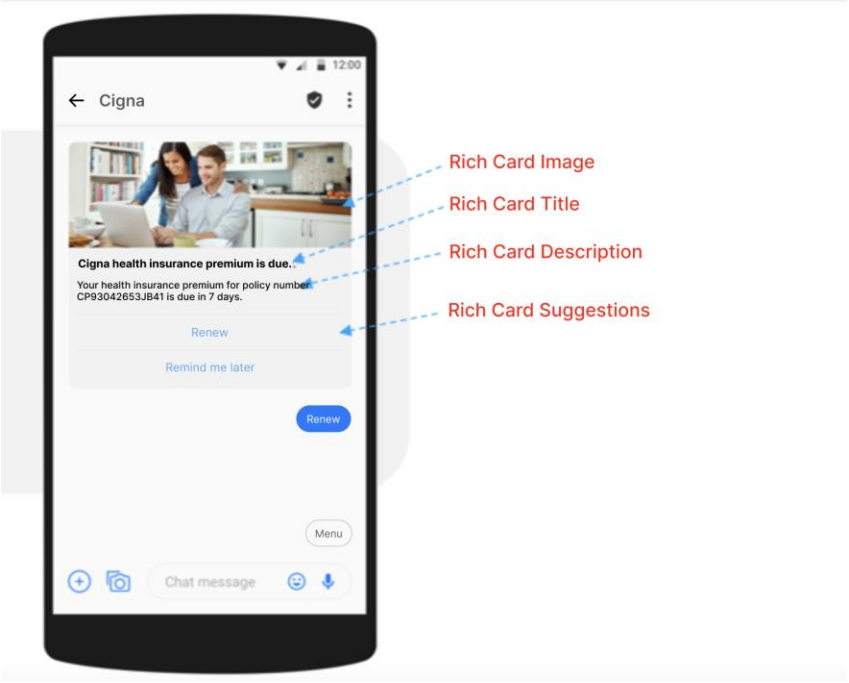
An RCS Template is a predefined set of RBM UI elements for e.g., a Rich Card with suggestions which can be used as a base for formulating messages in your campaign to the target audience. Once a bot is created, the aggregator can create multiple templates against the bot and use them for running campaigns. You can add images, videos, and suggested actions for your template.
Dotgo RBM portal supports 3 types of templates:

- Text Message
- Rich Card Standalone
- Rich Card Carousel

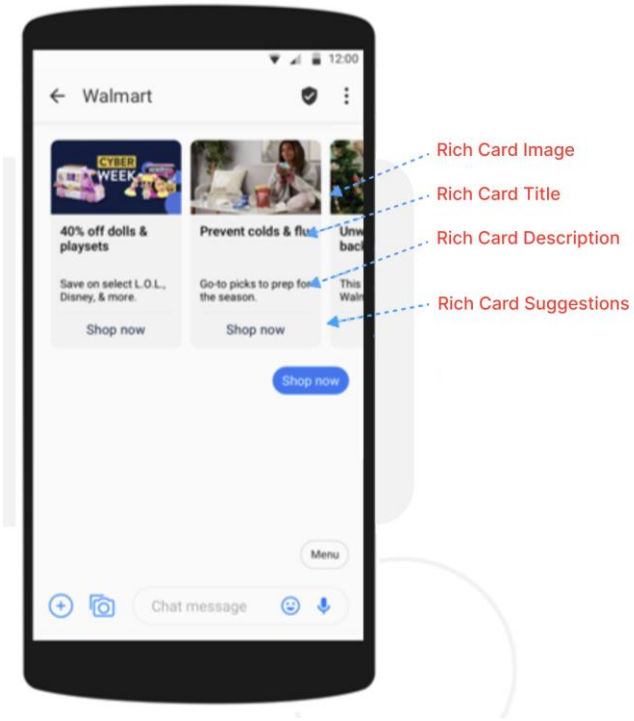
See figures below for examples of these template messages.



Text Message Template



Rich Card Standalone Template



Rich Card Carousel Template

5.2. Template Approval Process

Carriers in certain countries require the template to be approved before it can be used to send messages. Please contact Dotgo RBM Support at rbm-support@dotgo.com for more information about the Template Approval process.

6. API Throttling

The Dotgo RBM Platform defines configurations for the throttling limits for the API described in the previous sections. This means that there is an upper limit to the rate of API requests that the Dotgo RBM platform will accept from each account.

The Dtogo RBM Platform has two separate configurations for defining the throttling limits:

Group 1 Rate Limit

- This is defined in terms of API requests per second (TPS).
- Group 1 Rate Limit applies to following requests:
    - Individual Capability check
    - Sending a Message
    - Revoking a Message
- The TPS defined above is the average rate accepted by the Dotgo RBM Platform. In addition to that, the platform allows short time spikes of 20% of the configured TPS, as long as the load is below the configured TPS over a 30 second period.
- For example, if the TPS is set at 50, developer can send up to 60 TPS, as long as the total number of requests is no more than 1,500 in a 30 second window (50 TPS x 30 seconds = 1,500).

Group 2 Rate Limit

- This is defined in terms of API requests per minute (TPM).
- Group 2 Rate Limit applies to Bulk Capability Check

6.1. API Request Limits Exceeded

In the case that the number of requests for any given account or a given agent exceeds the throttling limit, the response object will be as below:

6.1.1. GSMA API

HTTP status - 429 Too Many Requests

```
{
    "RCSMessage": {
        "msgId": "b1ef710e-cc77-4b73-a3dc-d149dcae3aad",
        "status": "failed",
        "timestamp": "2023-03-07T10:42:50.615Z"
    },
    "reason":

{
        "text": "Number of API requests allowed per second exceeded. Please retry"

   }
}
```

6.1.2. Google API

HTTP status - 429 Too Many Requests

```
{
    "senderPhoneNumber": "+919686960276",
    "eventType": "FAILED",
    "reason": " Number of API requests allowed per second exceeded. Please retry"",

    "sendTime": "2023-03-07T10:45:37.959Z",
    "messageId": "e4d4529f-0336-4292-8a39-34964cad2bd3"
}
```

6.2. Rate Limits for your Account

Both Group 1 and Group 2 Rate Limits are configured and enforced at account level. Please contact rbm-support@dotgo.com for the value of TPS configured for your account.

7. Error Code Details

7.1. Sample error message for Http Status Code 401


```
{
   "RCSMessage": {
      "msgId": "0928aa1d-dce4-49d4-988c-1a8e031393e0",
      "status": "failed",
      "timestamp": "2024-11-14T14:57:57.828Z"
   },
   "reason": {
      "text": "Invalid ClientID",
      "code": “401-B”
   }
}
```


7.1.1. MaaP Specific Errors

For MaaP related issues following codes are used	Example of 400 error : 400-MXX
Google	M01

Dotgo	M02
JBM	M03
Jio	M04
Alaris	M05
Mavenir	M06
MavenirO2	M07
Orange	M08
Synchronoss	M09
Telcel	M10
TGS	M11
Vodafone	M12
Konnect	M13

7.2. HttpStatus Code 400 : BadRequest

Error Code	Error Message
400-A	Template provided with insufficient params
400-B	Template not approved
400-C	Test template limit exceeded
400-D	Template code with bot doesn't exist.
400-E	Agent Not found
400-F	Not a valid JSON message
400-G	Message type not supported
400-H	Invalid phone Number
400-MXX(to diff MaaPs) Ex : 400-M01 for google	400 error from MaaP reason/message(error extracted from MaaP) Format : error reason

7.3. HttpStatus Code 401 : Unauthorized

Error Code	Error Message
401-A	Unauthorized
401-B	Invalid ClientId
401-C	Invalid IP Address
401-D	User is not in conversation and provided message is not a template

7.4. HttpStatus Code 402 : Payment Required

Error Code	Error Message
402-A	Low balance. Please recharge

7.5. HttpStatus Code 403 : Forbidden

Error Code	Error Message
403-A	Bot not launched
403-B	User opted out
403-C	Bot blocked for spam

7.6. HttpStatus Code 404 : Not Found

Error Code	Error Message
404-A	Rcs disabled

7.7. HttpStatus Code 409 : Conflict

Error Code	Error Message
409-MXX(to diff MaaPs) Ex : 409-M01 for google	409 error from MaaP and will send error reason extracted from Particular MaaP

7.8. HttpStatus Code 429 : Too Many Requests

Error Code	Error Message
429-A	Too many requests
429-B	Monthly message limit exceeded
429-MXX Ex : 429-M01 for google	Too many requests error from downstream platforms/MaaPs.

7.9. HttpStatus Code 500 : Internal Server Error

Error Code	Error Message
500-A	Internal server error

7.10. HttpStatus Code 502 : Bad Gateway

Error Code	Error Message
502-A	Bad gateway
502-MXX Ex : 502-M01 for google	Timeout from Downstream MaaP

7.11. HttpStatus Code 503 : Service Unavailable

Error Code	Error Message

503-A	Curfew hours

7.12. HttpStatus Code 504 : Gateway Timeout

Error Code	Error Message
504-A	Timeout

8. Billing Events

Every time the message is updated on billing, an event is generated and sent to the aggregator to update on the state of the message and how it is charged. The events will be sent on agent webhook configured during creation. The message will be charged as 'A2P Single Message', 'Basic Message', 'A2P Conversation', 'P2A Message', 'P2A Conversation' or 'Trans 24'. These types are decided based on the flow of the messages. Additionally, events will be sent in cases where the message fails due to MaaP or gets revoked.

Primarily, there are 3 flows which lead to conversation:

A2P Flow :

Agent initiates the first message --> User responds --> Agent responds --> A2P Conversation

P2A Flow :

User initiates the first message --> Agent responds --> P2A Conversation

Trans 24 : (applicable only for approved Trans 24 agents)

Agent initiates the first message --> Agent sends second message --> Trans 24

Refund : Agent sends the message -> Message charged -> MaaP failure / revoked -> Refund

Request Parameter:

Name	Description
messageId	Unique Identifier for the event
billingEventId	New events will have messageId as billingEventId Update events will have first messageId which initiated the conversation as billingEventId with suffix(_conv)
billingEventAction	New event or Update event Values: New, Update
billingEventType	Type of the request being charged Values: 'A2P Single Message', 'Basic Message', 'A2P Conversation', 'P2A Message', 'P2A Conversation' or 'Trans 24'
billingCategory	Billing category. Possible values : A2P Single Message, P2A Message, Conversational, Basic Message, Trans 24
billingEventTime	Time of the billing event
agentId	Identifier of Agent
agentName	Agent name
userId	User phone number
countryId	Two-character country code
carrierId	Carrier Id (if available)

Please find below the example payloads that will be sent for each kind of message in each of the flows.

A2P Flow :

1. First A2P Message


```
{
  "messageId":"501b7644-7976-49d8-9fbf-4488238d2832",
  "billingEventId":"501b7644-7976-49d8-9fbf-4488238d2832",
  "billingEventAction":"New",
  "billingEventType":"A2P Single Message",
  "billingCategory":"A2P Single Message",
  "billingEventTime":"2024-02-26 16:42:00",
  "agentId": "YO4K83r3PhXsmY8m",
  "agentName": "Dummy bot",
  "userId": "+916362412935",
  "countryId": "IN",
  "carrierId": "404045"
```

```
}
```

2. Then First P2A Message

```
{
  "billingEventId":"401b7644-3067-5676-9fbf-4488000d2832",
  “messageId”:”401b7644-3067-5676-9fbf-4488000d2832”,
  "billingEventAction":"New",
  "billingEventType":"P2A Message",
  "billingCategory":"P2A Message",
  "billingEventTime":"2024-02-26 16:42:00"
  "agentId": "YO4K83r3PhXsmY8m",
  "agentName": "Dummy bot",
  "userId": "+916362412935",
  "countryId": "IN",
  "carrierId": "404045"
}
```

3. Then A2P message which will lead to A2P conv, here an Update Event is sent.

```
{
  "billingEventId":"501b7644-7976-49d8-9fbf-4488238d2832_conv", //first a2p billingEventId
with suffix conv.
   “messageId”:”04df2dcc-4166-4a82-9547-c3ef97ebd9e0”,
  "billingEventIdP2A":"401b7644-3067-5676-9fbf-4488000d2832",
  "billingEventAction":"Update",
  "billingEventType":"A2P Conversation",
  "billingCategory":"Conversational",
  "billingEventTime":"2024-02-26 16:42:00"
  "agentId": "YO4K83r3PhXsmY8m",
  "agentName": "Dummy bot",
  "userId": "+916362412935",
  "countryId": "IN",
  "carrierId": "404045"
}
```

4. If any A2P or P2A message is sent within the window, following json will be sent.

a. Incase of a2p

```
 {
   "billingEventId":"501b7644-7976-49d8-9fbf-4488238d2832_conv", //first a2p
billingEventId
   “messageId”:”12f7d79d-9470-472c-a8c9-476db92a7b85”,
   "billingEventType":"A2P Conversation",
   "billingCategory":"Conversational",
   "billingEventTime":"2024-02-26 16:42:00"
   "agentId": "YO4K83r3PhXsmY8m",
   "agentName": "Dummy bot",
   "userId": "+916362412935",
   "countryId": "IN",
   "carrierId": "404045"
 }
```

b. Incase of p2a


```
{
  "billingEventId":"501b7644-7976-49d8-9fbf-4488238d2832_conv", //first a2p billingEventId
  “messageId”:”963d7d0d-a9ba-42ed-9eca-c314c7777d35”,
  "billingEventType":"A2P Conversation",
  "billingCategory":"Conversational",
  "billingEventTime":"2024-02-26 16:42:00"
  "agentId": "YO4K83r3PhXsmY8m",
  "agentName": "Dummy bot",
  "userId": "+916362412935",
  "countryId": "IN",
  "carrierId": "404045"

}
```


P2A Flow :

1. First a P2A message is sent.


```
{
  "billingEventId":"401b7644-3067-5676-9fbf-4488000d2832",
  “messageId”:”401b7644-3067-5676-9fbf-4488000d2832”,
  "billingEventAction":"New",
  "billingEventType":"P2A Message",
  "billingCategory":"P2A Message",
  "billingEventTime":"2024-02-26 16:42:00"
  "agentId": "YO4K83r3PhXsmY8m",
  "agentName": "Dummy bot",
  "userId": "+916362412935",
  "countryId": "IN",
  "carrierId": "404045"
}
```


2. Then A2P message is sent which will lead to P2A conversation. And an UpdateEvent is sent.


```
{
  "billingEventId":"401b7644-3067-5676-9fbf-4488000d2832_conv", //first p2a billingEventId with suffix conv.
  “messageId”:”963d7d0d-a9ba-42ed-9eca-c314c7777d35”,
  "billingEventAction":"Update",
  "billingEventType":"P2A Conversation",
  "billingCategory":"Conversational",
  "billingEventTime":"2024-02-26 16:43:00"
  "agentId": "YO4K83r3PhXsmY8m",
  "agentName": "Dummy bot",
  "userId": "+916362412935",
  "countryId": "IN",
  "carrierId": "404045"
}
```

3. If any A2P or P2A message is sent within the window. Following json will be sent.


```
{
 "billingEventId":"401b7644-3067-5676-9fbf-4488000d2832_conv", //first p2a billingEventId with suffix conv.
 “messageId”:”2190930d-119c-4f09-b5f3-09b8ea4fef81”,
 "billingEventType":"P2A Conversation",
 "billingCategory":" Conversational",
 "billingEventTime":"2024-02-26 16:43:00"
 "agentId": "YO4K83r3PhXsmY8m",
 "agentName": "Dummy bot",
  "userId": "+916362412935",
 "countryId": "IN",
 "carrierId": "404045"
}
```


Trans 24 :

1. First A2P Message


```
{
 "billingEventId":"9081644-3067-5676-9fbf-4488000d2832",
 “messageId”: “9081644-3067-5676-9fbf-4488000d2832”,
 "billingEventAction":"New",
 "billingEventType":"A2P Single message",
 "billingCategory":"A2P Single Message",
 "billingEventTime":"2024-02-26 16:42:00"
 "agentId": "YO4K83r3PhXsmY8m",
 "agentName": "Dummy bot",
 "userId": "+916362412935",
 "countryId": "IN",
 "carrierId": "404045"
}
```


2. Second A2P Message


```
{
 "billingEventId":"9081644-3067-5676-9fbf-4488000d2832_conv", // conversation
 “messageId”: “124244-3067-5676-9fbf-4488000d2832”,
 "billingEventAction":"Update"
 "billingEventType":"Trans 24",
 "billingCategory":" Trans 24",
 "billingEventTime":"2024-02-26 16:42:10"
 "agentId": "YO4K83r3PhXsmY8m",
 "agentName": "Dummy bot",
 "userId": "+916362412935",
 "countryId": "IN",
 "carrierId": "404045"
}
```


3. P2A Message

```
{
  "billingEventId":"9081644-3067-5676-9fbf-4488000d2832_conv",
  “messageId”: “5783644-3067-5676-9fbf-4488000d2832”,
  "billingEventType":"Trans 24",
  "billingCategory":" Trans 24",
  "billingEventTime":"2024-02-26 16:42:20"
  "agentId": "YO4K83r3PhXsmY8m",
  "agentName": "Dummy bot",
  "userId": "+916362412935",
  "countryId": "IN",
  "carrierId": "404045"
}
```


Refund Event:


```
{
   "message_id": "4683644-3067-5676-9fbf-4488000d2532",
   "billing_event_id": "4683644-3067-5676-9fbf-4488000d2532",
   "billing_event_action": "refund_complete",
   "billing_category": "basic_message",
   "billing_event_type": "basic_message",
   "billing_event_time": "2024-07-12 14:39:30",
   "agent_id": "YO4K83r3PhXsmY8m",
   "agent_name": "Dominos",
   "user_id": "917774777132",
   "country_id": "IN"
}
```