Viber API Documentation 7.3.0

Viber REST API

Table of Contents

Get Started

In order to implement the API you will need the following:

1. **An Active Viber account** on a platform which supports Public Accounts/ bots (iOS/Android). This account will automatically be set as the account administrator during the account creation process.
2. **Active Public Account/ bot** - Create an account here.

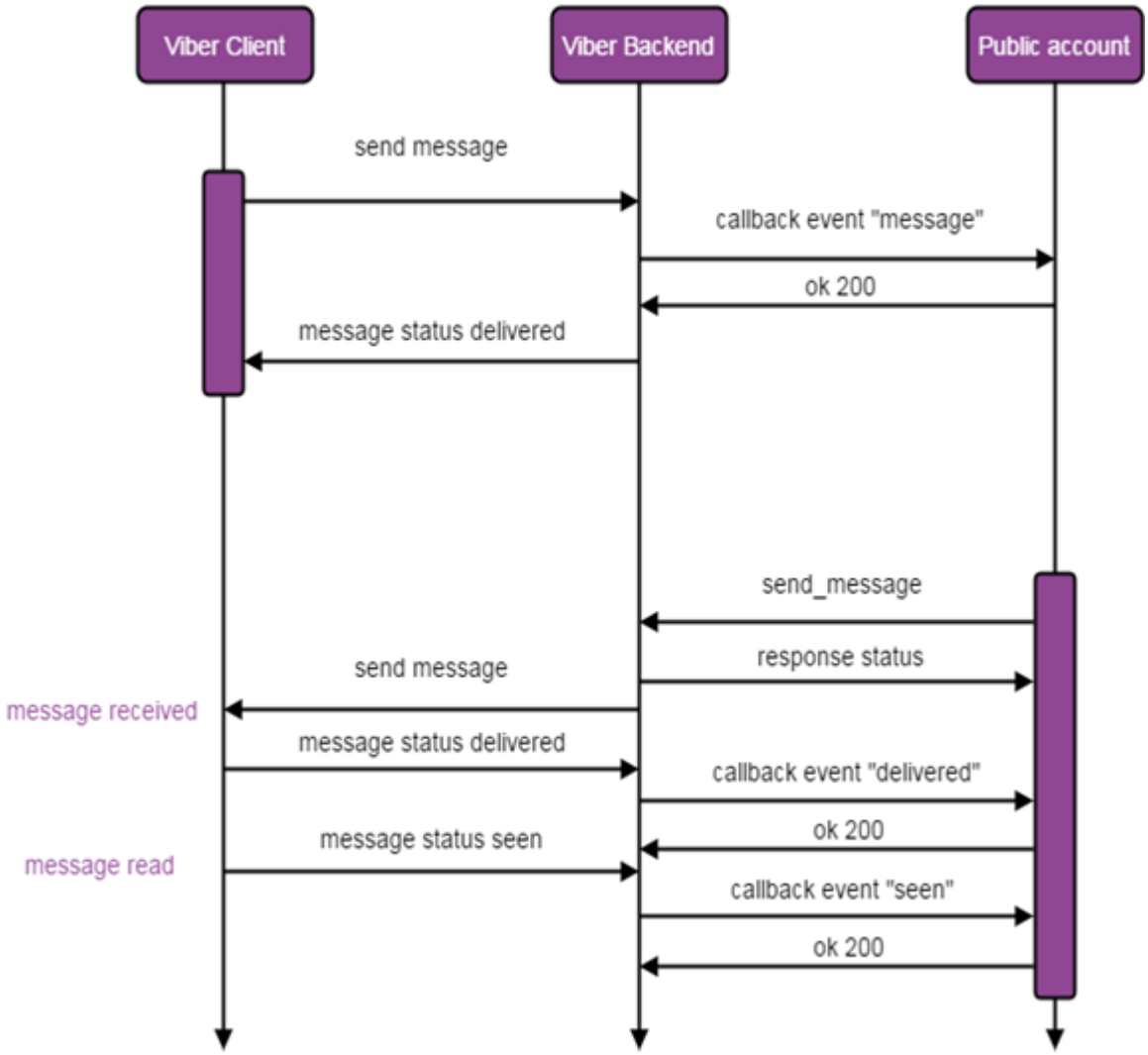
3. **Account authentication token** - unique account identifier used to validate your account in all API requests. Once your account is created your authentication token will appear in the account's "edit info" screen (for admins only). Each request posted to Viber by the account will need to contain the token.
4. **Setup account webhook** – this needs to be done once during the account setup process to define your webhook and the type of responses you would like to receive. In order to implement the API you will need the following:

Supported platforms

Public Accounts/ bots are currently supported on iOS and Android devices running Viber version 6.5 and above and on desktop from version 6.5.3.

Send and receive message flow

The following diagram describes the flow of sending and receiving messages by the account. All API requests and callbacks mentioned in the diagram will be explained later in this document.



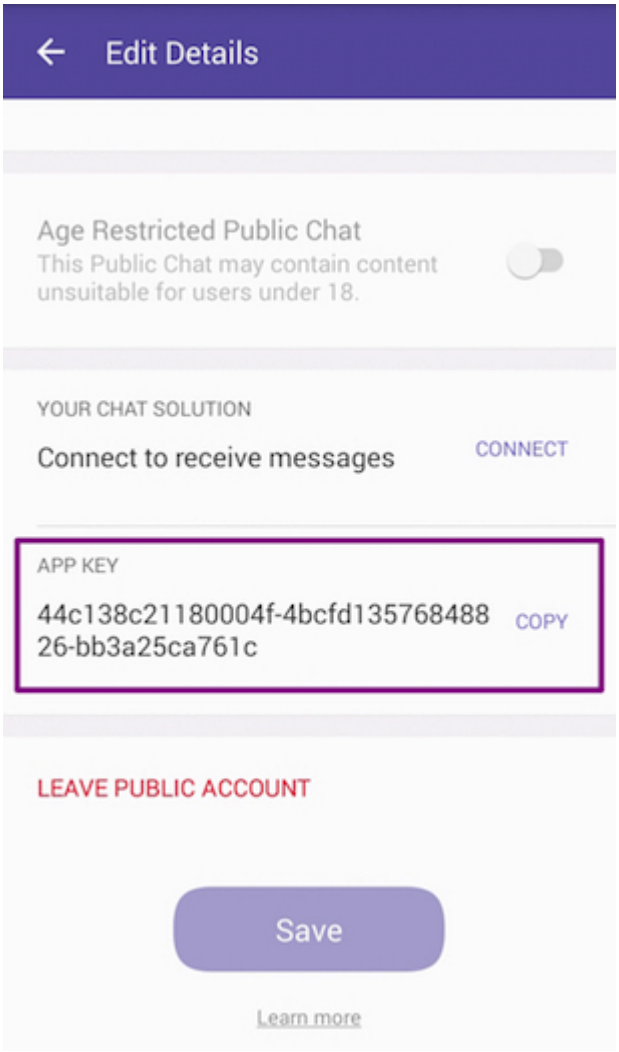
Authentication token

The authentication token (also known as application key) is a unique and secret account identifier. It is used to authenticate request in the Viber API and to prevent unauthorized persons from sending requests on behalf of a Public Account/ bot. Each API request must include an HTTP Header called `X-Viber-Auth-Token` containing the account's authentication token.

HTTP header

```
X-Viber-Auth-Token: 445da6az1s345z78-dazcczb2542zv51a-e0vc5fva17480im9
```

The authentication token is generated upon account creation and can be viewed by the account's admins in the "edit info" screen of their Public Account/ bot or on the Viber Admin Panel.



Note: Failing to send the authentication token in the header will result in an error with the `missing_auth_token` message.

Webhooks

Once you have your token you will be able to set your account's webhook. This webhook will be used for receiving callbacks and user messages from Viber.

Setting the webhook will be done by calling the `set_webhook` API with a valid & certified URL. For security reasons only URLs with valid and official SSL certificate from a trusted CA will be allowed. This action defines the account's webhook and the type of events the account wants to be notified about.

Once a `set_webhook` request is sent Viber will send a callback to the webhook to check its availability and return a response to the user.

Note that once you set your webhook the 1-on-1 conversation with your account will become available. To disable 1-on-1 conversation with your account you'll need to remove your webhook – see removing your webhook below.

Viber's API allows you to receive user names and photos. This has been updated to comply with privacy laws and allow developers who don't make use of user names and photos as part of their service to opt out. If you don't use user names photos, please opt-out to default values.

To set the request, pass `send_name` and `send_photo` flags with the `set_webhook` request.

Note: This feature will work if the user has allowed "Content Personalisation" (More → Privacy → personal data). If the user has disallowed content personalization, you will receive placeholder values.

Setting a Webhook

Resource URL

```
https://chatapi.viber.com/pa/set_webhook
```

Post data

```
{
   "url":"https://my.host.com",
   "event_types":[
      "delivered",
      "seen",
      "failed",
      "subscribed",
```

```
        "unsubscribed",
        "conversation_started"
    ],
    "send_name": true,
    "send_photo": true
}
```

Post parameters

Name	Description	Validation
url	required. Account webhook URL to receive callbacks & messages from users	Webhook URL must use SSL Note: Viber doesn't support self signed certificates
event_types	optional. Indicates the types of Viber events that the account owner would like to be notified about. **Don't include this parameter in your request to get all events**	Possible values: `delivered` , `seen` , `failed` , `subscribed` , `unsubscribed` and `conversation_started`
send_name	optional. Indicates whether or not the bot should receive the user name. Default `false`	Possible values: `true` , `false`
send_photo	optional. Indicates whether or not the bot should receive the user photo. Default `false`	Possible values: `true` , `false`

Set webhook Response

```
{
    "status":0,
    "status_message":"ok",
    "event_types":[
        "delivered",
        "seen",
        "failed",
        "subscribed",
        "unsubscribed",
        "conversation_started"
    ]
}
```

Response parameters

Name	Description	Possible values

Name	Description	Possible values
status	Action result	`0` for success. In case of failure – appropriate failure status number. See error codes table for additional information
status_message	`ok` or failure reason	Success: `ok` . Failure: `invalidUrl` , `invalidAuthToken` , `badData` , `missingData` and `failure` . See error codes table for additional information
event_types	List of event types you will receive a callback for. Should return the same values sent in the request	`delivered` , `seen` , `failed` , `subscribed` , `unsubscribed` and `conversation_started`

Event types filtering

The `event_types` parameter allows accounts to choose which events they would get a callback for. The following events are mandatory and can't be filtered out: `message` , `subscribed` and `unsubscribed` .
The following events can be filtered out during the `set_webhook` request: `delivered` , `seen` , `failed` and `conversation_started` .
Sending a `set_webhook` request with no event_types parameter means getting **all** events.
Sending a `set_webhook` request with empty `event_types` list `("event_types": [])` means getting **only mandatory** events. See callbacks section for full callbacks events information.

Set webhook callback

For each `set_webhook` request Viber will send a callback to the webhook URL to confirm it is available. The expected HTTP response to the callback is `200` OK – any other response will mean the webhook is not available. If the webhook is not available the `set_webhook` response sent to the user will be status 1: invalidUrl.

Callback data

```
{
   "event":"webhook",
   "timestamp":1457764197627,
   "message_token":241256543215
}
```

Callback parameters

Name	Description	Possible values
event	Callback type – which event triggered the callback	webhook

Name	Description	Possible values
timestamp	Time of the event that triggered the callback	Epoch time
message_token	Unique ID of the message	

Removing your webhook

Once you set a webhook to your Public Account/ bot your 1-on-1 conversation button will appear and users will be able to access it.
At the moment there is no option to disable the 1-on-1 conversation from the Public Account/ bot settings, so to disable this option you'll need to remove the webhook you set for the account.
Removing the webhook is done by Posting a `set_webhook` request with an empty webhook string.

Resource URL

```
https://chatapi.viber.com/pa/set_webhook
```

Post data

```
{
   "url":""
}
```

Post parameters

Name	Description
url	required. Account webhook URL to receive callbacks & messages from users. In this case, use an empty string to remove any previously set webhook

Send Message

The `send_message` API allows accounts to send messages to Viber users who subscribe to the account. Sending a message to a user will be possible only after the user has subscribed to the Public Account by pressing the subscribe button or by sending a message, or by sending a message to a bot (see `subscribed` callback for additional information).

The API supports a variety of message types: `text` , `picture` , `video` , `file` , `location` , `sticker` , `contact` , `carousel content` and `URL` . Specific post data examples and required parameters for each message type are given below.

Validation

Maximum total JSON size of the request is 30kb.

Resource URL

```
https://chatapi.viber.com/pa/send_message
```

General send message parameters

The following parameters are available for all message types:

Name	Description	Validation
receiver	Unique Viber user id	required, subscribed valid user id
type	Message type	required. Available message types: `text`, `picture`, `video`, `file`, `location`, `contact`, `sticker`, `carousel content` and `url`
sender.name	The sender's name to display	required. Max 28 characters
sender.avatar	The sender's avatar URL	optional. Avatar size should be no more than 100 kb. Recommended 720x720
tracking_data	Allow the account to track messages and user's replies. Sent `tracking_data` value will be passed back with user's reply	optional. max 4000 characters
min_api_version	Minimal API version required by clients for this message (default 1)	optional. client version support the API version

Message types

Below is a list of all supported message types with post data examples.

Text message

Post data

```
{
   "receiver":"01234567890A=",
   "min_api_version":1,
   "sender":{
      "name":"John McClane",
      "avatar":"http://avatar.example.com"
```

```
    },
    "tracking_data":"tracking data",
    "type":"text",
    "text":"Hello world!"
}
```

Post parameters

Name	Description	Validation
type	Message type	required. `text`
text	The text of the message	required

Picture message

Post data

```
{
    "receiver":"01234567890A=",
    "min_api_version":1,
    "sender":{
       "name":"John McClane",
       "avatar":"http://avatar.example.com"
    },
    "tracking_data":"tracking data",
    "type":"picture",
    "text":"Photo description",
    "media":"http://www.images.com/img.jpg",
    "thumbnail":"http://www.images.com/thumb.jpg"
}
```

Post parameters

Name	Description	Validation
type	Message type	required. `picture`
text	Description of the photo. Can be an empty string if irrelevant	required. Max 120 characters
media	URL of the image (JPEG)	required. Max size 1 MB. Only JPEG format is supported. Other image formats as well as animated GIFs can be sent as URL messages or file messages

Name	Description	Validation
thumbnail	URL of a reduced size image (JPEG)	optional. Max size 100 kb. Recommended: 400x400. Only JPEG format is supported

Video message

Post data

```
{
   "receiver":"01234567890A=",
   "min_api_version":1,
   "sender":{
      "name":"John McClane",
      "avatar":"http://avatar.example.com"
   },
   "tracking_data":"tracking data",
   "type":"video",
   "media":"http://www.images.com/video.mp4",
   "thumbnail":"http://www.images.com/thumb.jpg",
   "size":10000,
   "duration":10
}
```

Post parameters

Name	Description	Validation
type	Message type	required. `video`
media	URL of the video (MP4, H264)	required. Max size 50 MB. Only MP4 and H264 are supported
size	Size of the video in bytes	required
duration	Video duration in seconds; will be displayed to the receiver	optional. Max 180 seconds
thumbnail	URL of a reduced size image (JPEG)	optional. Max size 100 kb. Recommended: 400x400. Only JPEG format is supported

File message

Post data

```
{
   "receiver":"01234567890A=",
```

```
   "min_api_version":1,
   "sender":{
      "name":"John McClane",
      "avatar":"http://avatar.example.com"
   },
   "tracking_data":"tracking data",
   "type":"file",
   "media":"http://www.images.com/file.doc",
   "size":10000,
   "file_name":"name_of_file.doc"
}
```

Post parameters

Name	Description	Validation
type	Message type	required. `file`
media	URL of the file	required. Max size 50 MB. See forbidden file formats for unsupported file types
size	Size of the file in bytes	required
file_name	Name of the file	required. File name should include extension. Max 256 characters (including file extension). Sending a file without extension or with the wrong extension might cause the client to be unable to open the file

Contact message

Post data

```
{
   "receiver":"01234567890A=",
   "min_api_version":1,
   "sender":{
      "name":"John McClane",
      "avatar":"http://avatar.example.com"
   },
   "tracking_data":"tracking data",
   "type":"contact",
   "contact":{
      "name":"Itamar",
      "phone_number":"+972511123123"
   }
}
```

Post parameters

Name	Description	Validation
type	Message type	required. `contact`
contact.name	Name of the contact	required. Max 28 characters
contact.phone_number	Phone number of the contact	required. Max 18 characters

Location message

Post data

```
{
   "receiver":"01234567890A=",
   "min_api_version":1,
   "sender":{
      "name":"John McClane",
      "avatar":"http://avatar.example.com"
   },
   "tracking_data":"tracking data",
   "type":"location",
   "location":{
      "lat":"37.7898",
      "lon":"-122.3942"
   }
}
```

Post parameters

Name	Description	Validation
type	Message type	required. `location`
location	Location coordinates	required. latitude (±90°) & longitude (±180°) within valid ranges

URL message

Post data

```
{
   "receiver":"01234567890A=",
   "min_api_version":1,
   "sender":{
      "name":"John McClane",
```

```
      "avatar":"http://avatar.example.com"
   },
   "tracking_data":"tracking data",
   "type":"url",
   "media":"http://www.website.com/go_here"
}
```

Post parameters

Name	Description	Validation
type	Message type	required. `url`
media	URL	required. Max 2,000 characters

Sticker message

Post data

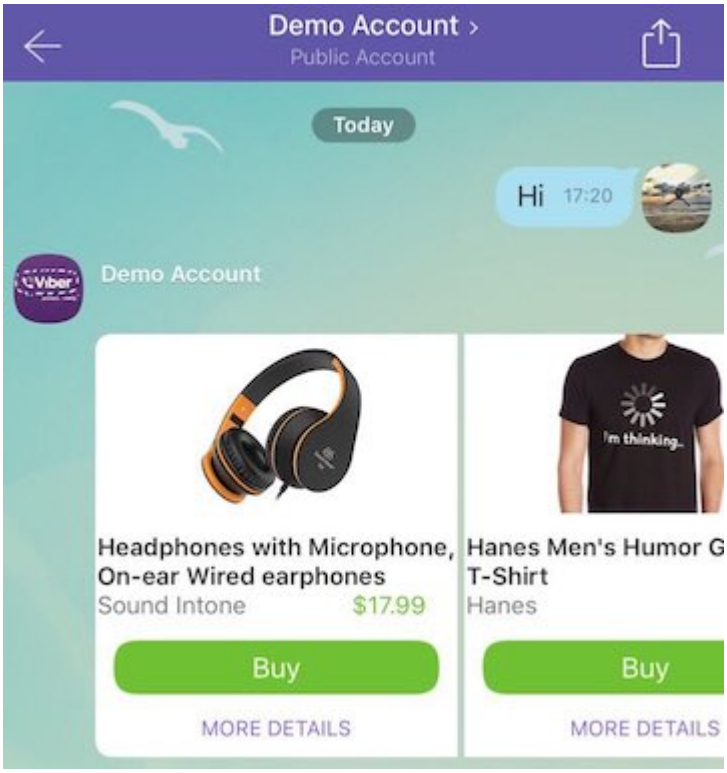
```
{
   "receiver":"01234567890A=",
   "min_api_version":1,
   "sender":{
      "name":"John McClane",
      "avatar":"http://avatar.example.com"
   },
   "tracking_data":"tracking data",
   "type":"sticker",
   "sticker_id":46105
}
```

Post parameters

Name	Description	Validation
type	Message type	required. `sticker`
sticker_id	Unique Viber sticker ID. For examples visit the sticker IDs page	

Carousel content message

The Carousel Content Message type allows a user to scroll through a list of items, each composed of an image, description and call to action button.



Note: Carousel Content Message is supported on devices running Viber version 6.7 and above.

Post data


```
{
   "receiver":"nsId6t9MWy3mq09RAeXiug==",
   "type":"rich_media",
   "min_api_version":2,
   "rich_media":{
      "Type":"rich_media",
      "ButtonsGroupColumns":6,
      "ButtonsGroupRows":7,
      "BgColor":"#FFFFFF",
      "Buttons":[
         {
            "Columns":6,
            "Rows":3,
            "ActionType":"open-url",
            "ActionBody":"https://www.google.com",
            "Image":"http://html-test:8080/myweb/guy/assets/imageRMsmall2.png"
         },
         {
            "Columns":6,
            "Rows":2,
            "Text":"<font color=#323232><b>Headphones with Microphone, On-ear Wired e
```

```
            "ActionType":"open-url",
            "ActionBody":"https://www.google.com",
            "TextSize":"medium",
            "TextVAlign":"middle",
            "TextHAlign":"left"
        },
        {
            "Columns":6,
            "Rows":1,
            "ActionType":"reply",
            "ActionBody":"https://www.google.com",
            "Text":"<font color=#ffffff>Buy</font>",
            "TextSize":"large",
            "TextVAlign":"middle",
            "TextHAlign":"middle",
            "Image":"https://s14.postimg.org/4mmt4rw1t/Button.png"
        },
        {
            "Columns":6,
            "Rows":1,
            "ActionType":"reply",
            "ActionBody":"https://www.google.com",
            "Text":"<font color=#8367db>MORE DETAILS</font>",
            "TextSize":"small",
            "TextVAlign":"middle",
            "TextHAlign":"middle"
        },
        {
            "Columns":6,
            "Rows":3,
            "ActionType":"open-url",
            "ActionBody":"https://www.google.com",
            "Image":"https://s16.postimg.org/wi8jx20wl/image_RMsmall2.png"
        },
        {
            "Columns":6,
            "Rows":2,
            "Text":"<font color=#323232><b>Hanes Men's Humor Graphic T-Shirt</b></fon
            "ActionType":"open-url",
            "ActionBody":"https://www.google.com",
            "TextSize":"medium",
            "TextVAlign":"middle",
```

```
            "TextHAlign":"left"
         },
         {
            "Columns":6,
            "Rows":1,
            "ActionType":"reply",
            "ActionBody":"https://www.google.com",
            "Text":"<font color=#ffffff>Buy</font>",
            "TextSize":"large",
            "TextVAlign":"middle",
            "TextHAlign":"middle",
            "Image":"https://s14.postimg.org/4mmt4rw1t/Button.png"
         },
         {
            "Columns":6,
            "Rows":1,
            "ActionType":"reply",
            "ActionBody":"https://www.google.com",
            "Text":"<font color=#8367db>MORE DETAILS</font>",
            "TextSize":"small",
            "TextVAlign":"middle",
            "TextHAlign":"middle"
         }
      ]
   }
}
```

Post parameters

Name	Description	Possible values
alt_text	Backward compatibility text, limited to 7,000 characters	
rich_media.ButtonsGroupColumns	Number of columns per carousel content block. Default 6 columns	1 - 6
rich_media.ButtonsGroupRows	Number of rows per carousel content block. Default 7 rows	1 - 7
rich_media.Buttons	Array of buttons	Max of 6 * ButtonsGroupColumns * ButtonsGroupRows

Button element

Name	Description	Possible values
Columns	Button column span. Default ButtonsGroupColumns	1..ButtonsGroupColumns
Rows	Button row span. Default ButtonsGroupRows	1..ButtonsGroupRows

Keyboards

The Viber API allows sending a custom keyboard using the `send_message` API, to supply the user with a set of predefined replies or actions. The keyboard can be attached to any message type or sent on it's on. Once received, the keyboard will appear to the user instead of the device's native keyboard. The keyboards are fully customizable and can be created and designed specifically for the account's needs. The client will always display the last keyboard that was sent to it.

Read the following article to learn more about keyboards.

Validation

Maximum total JSON size of the request is 30kb.

Resource URL

```
https://chatapi.viber.com/pa/send_message
```

Post data

Keyboards can be attached to any message type and be sent and displayed together. To attach a keyboard to a message simply add the keyboard's parameters to the message JSON.
The example below shows a keyboard sent with a text message (note that the keyboard doesn't contain all optional values).

```
{
   "receiver":"01234567890A=",
   "type":"text",
   "text":"Hello world",
   "keyboard":{
      "Type":"keyboard",
      "DefaultHeight":true,
      "Buttons":[
         {
```

```
            "ActionType":"reply",
            "ActionBody":"reply to me",
            "Text":"Key text",
            "TextSize":"regular"
         }
      ]
   }
}
```

Broadcast Message

The `broadcast_message` API allows accounts to send messages to multiple Viber users who subscribe to the account. Sending a message to a user will be possible only after the user has subscribed to the Public Account by pressing the subscribe button or by sending a message, or by sending a message to a bot. The ability to send broadcast messages is only opened on application and approval from Viber account managers.

The API supports a variety of message types: `text` , `picture` , `video` , `file` , `location` , `sticker` , `contact` , `carousel content` and `URL` .

Validation

Maximum total JSON size of the request is 30kb. The maximum list length is **300** receivers. The Broadcast API is used to send messages to multiple recipients with a rate limit of 500 requests in a 10 seconds window.

Resource URL

```
https://chatapi.viber.com/pa/broadcast_message
```

Post parameters

This API method uses the same parameters as the `send` REST API method with a few variations described below.

broadcast_list

This mandatory parameter defines the recipients for the message. Every user must be subscribed and have a valid user id. The maximum list length is 300 receivers. For example (this should be a part of the full JSON body):

```
{
   "broadcast_list":[
      "ABB102akPCRKFaqxWnafEIA==",
      "ABB102akPCRKFaqxWna111==",
      "ABB102akPCRKFaqxWnaf222=="
   ]
}
```

Place holders

Broadcast message can contain place holders that will be replaced with receiver information (each receiver will get it's own information). The place holders can appear anywhere in the message, even in tracking data. The list of the place holders:

- `replace_me_with_receiver_id` - will be replaced by the receiver ID
- `replace_me_with_url_encoded_receiver_id` - will be replaced by the URL encoded receiver ID
- `replace_me_with_user_name` - will be replaced by the receiver user name

Post example

The following example demonstrates send carousel content with place holders (`replace_me_with_receiver_id` , `replace_me_with_url_encoded_receiver_id` , `replace_me_with_user_name`) to 4 receivers:

```
{
   "sender":{
      "name":"John McClane",
      "avatar":"http://avatar.example.com"
   },
   "min_api_version":2,
   "type":"rich_media",
   "broadcast_list":[
      "pttm25kSGUo1919sBORWyA==",
      "2yBSIsbzs7sSrh4oLm2hdQ==",
      "EGAZ3SZRi6zW1D0uNYhQHg==",
      "kBQYX9LrGyF5mm8JTxdmpw=="
   ],
   "rich_media":{
      "Type":"rich_media",
      "BgColor":"#FFFFFF",
      "Buttons":[
```

```
        {
           "ActionBody":"https://www.google.com",
           "ActionType":"open-url",
           "Text":"Should get back my ID instead of replace_me_with_receiver_id"
        },
        {
           "ActionBody":"https://www.google.com",
           "ActionType":"open-url",
           "Text":"Should get back my URL encoded ID instead of replace_me_with_url_
        },
        {
           "ActionBody":"https://www.google.com",
           "ActionType":"open-url",
           "Text":"Should get back my name instead of replace_me_with_user_name"
        }
     ]
  }
}
```

Response

Response parameters

Name	Description	Possible values
message_token	Unique ID of the message	
status	Action result	`0` for success. In case of failure – appropriate failure status number. See error codes table for additional information
status_message	`ok` or failure reason	Success: `ok` . Failure: `invalidUrl` , `invalidAuthToken` , `badData` , `missingData` and `failure` . See error codes table for additional information
failed_list	Contains all the receivers to which the message could not be sent properly	See error codes table for additional information

Response example

```
{
   "message_token":40808912438712,
   "status":0,
   "status_message":"ok",
```

```
   "failed_list":[
      {
         "receiver":"pttm25kSGUo1919sBORWyA==",
         "status":6,
         "status_message":"Not subscribed"
      },
      {
         "receiver":"EGAZ3SZRi6zW1D0uNYhQHg==",
         "status":5,
         "status_message":"Not found"
      }
   ]
}
```

Get Account Info

The `get_account_info` request will fetch the account's details as registered in Viber. The account admin will be able to edit most of these details from his Viber client.

Resource URL

```
https://chatapi.viber.com/pa/get_account_info
```

Post data

```
{
}
```

Response

```
{
   "status":0,
   "status_message":"ok",
   "id":"pa:75346594275468546724",
   "name":"account name",
   "uri":"accountUri",
   "icon":"http://example.com",
```

```
    "background":"http://example.com",
    "category":"category",
    "subcategory":"sub category",
    "location":{
       "lon":0.1,
       "lat":0.2
    },
    "country":"UK",
    "webhook":"https://my.site.com",
    "event_types":[
       "delivered",
       "seen"
    ],
    "subscribers_count":35,
    "members":[
       {
          "id":"01234567890A=",
          "name":"my name",
          "avatar":"http://example.com",
          "role":"admin"
       }
    ]
}
```

Response parameters

Name	Description	Possible values
status	Action result	`0` for success. In case of failure – appropriate failure status number. See error codes table for additional information
status_message	`ok` or failure reason	Success: `ok` . Failure: `invalidUrl` , `invalidAuthToken` , `badData` , `missingData` and `failure` . See error codes table for additional information
id	Unique numeric id of the account	
name	Account name	Max 75 characters
uri	Unique URI of the Account	
icon	Account icon URL	JPEG, 720x720, size no more than 512 kb
background	Conversation background URL	JPEG, max 1920x1920, size no more than 512 kb

Name	Description	Possible values
category	Account category	
subcategory	Account sub-category	
location	Account location (coordinates). Will be used for finding accounts near me	lat & lon coordinates
country	Account country	2 letters country code - ISO ALPHA-2 Code
webhook	Account registered webhook	webhook URL
event_types	Account registered events – as set by `set_webhook` request	`delivered` , `seen` , `failed` and `conversation_started`
subscribers_count	Number of subscribers	
members	Members of the Public Account's public chat	`id` , `name` , `avatar` , `role` for each Public Chat member (admin/participant). Public Accounts only

Get User Details

The `get_user_details` request will fetch the details of a specific Viber user based on his unique user ID. The user ID can be obtained from the callbacks sent to the account regarding user's actions. This request can be sent twice during a 12 hours period for each user ID.

Resource URL

```
https://chatapi.viber.com/pa/get_user_details
```

Post data

```
{
   "id":"01234567890A="
}
```

Post parameters

Name	Description	Validation
id	Unique Viber user id	required. subscribed valid user id

Response


```
{
   "status":0,
   "status_message":"ok",
   "message_token":4912661846655238145,
   "user":{
      "id":"01234567890A=",
      "name":"John McClane",
      "avatar":"http://avatar.example.com",
      "country":"UK",
      "language":"en",
      "primary_device_os":"android 7.1",
      "api_version":1,
      "viber_version":"6.5.0",
      "mcc":1,
      "mnc":1,
      "device_type":"iPhone9,4"
   }
}
```


Response parameters

Name	Description	Possible values
status	Action result	`0` for success. In case of failure – appropriate failure status number. See error codes table for additional information
status_message	`ok` or failure reason	Success: `ok` . Failure: `invalidUrl` , `invalidAuthToken` , `badData` , `receiverNoSuitableDevice` , `missingData` and `failure` . See error codes table for additional information
message_token	Unique id of the message	
user.id	Unique Viber user id	
user.name	User's Viber name	
user.avatar	URL of the user's avatar	
user.country	User's country code	2 letters country code - ISO ALPHA-2 Code
user.language	User's phone language. Will be returned according to the device language	ISO 639-1

Name	Description	Possible values
user.primary_device_os	The operating system type and version of the user's primary device.	
user.api_version	Max API version, matching the most updated user's device	Currently only 1. Additional versions will be added in the future
user.viber_version	The Viber version installed on the user's primary device	
user.mcc	Mobile country code	
user.mnc	Mobile network code	
user.device_type	The user's device type	

Get Online

The `get_online` request will fetch the online status of a given subscribed account members. The API supports up to 100 user id per request and those users must be subscribed to the account.

Resource URL

```
https://chatapi.viber.com/pa/get_online
```

Post data

```
{
   "ids":[
      "01234567890=",
      "01234567891=",
      "01234567893="
   ]
}
```

Post parameters

Name	Description	Validation

Name	Description	Validation
ids	Unique Viber user id	required. 100 ids per request

Response

```
{
   "status":0,
   "status_message":"ok",
   "users":[
      {
         "id":"01234567890=",
         "online_status":0,
         "online_status_message":"online"
      },
      {
         "id":"01234567891=",
         "online_status":1,
         "online_status_message":"offline",
         "last_online":1457764197627
      },
      {
         "id":"01234567893=",
         "online_status":3,
         "online_status_message":"tryLater"
      }
   ]
}
```

Response parameters

Name	Description	Possible values
status	Action result	`0` for success. In case of failure – appropriate failure status number. See error codes table for additional information
status_message	`ok` or failure reason	Success: `ok` . Failure: `invalidUrl` , `invalidAuthToken` , `badData` , `missingData` and `failure` . See error codes table for additional information
user[x].id	Unique Viber user id	

Name	Description	Possible values
user[x].online_status	Online status code	`0` for online, `1` for offline, `2` for undisclosed - user set Viber to hide status, `3` for try later - internal error, `4` for unavailable - not a Viber user / unsubscribed / unregistered
user[x]. online_status_message	Online status message	

Post to Public Chat

The post API allows the Public Account owner to post a message in the Public Account's public chat.

Note: The post API is fully supported on Android and iOS versions 6.5.3 and above. Versions 6.5 and below support the post API for text and link messages only. Other message types (picture, video, etc.) will be converted and displayed as a link leading to the message's media. Support for the post API on Viber's desktop version is currently in beta phase and may not be fully compatible with all message types.

Validation

Maximum total JSON size of the request is 30kb.

Resource URL

```
https://chatapi.viber.com/pa/post
```

Post data

The following example shows a post request of a text message. For other message types see possible message types below.

```
{
   "from":"01234567890A=",
   "sender":{
      "name":"John McClane",
      "avatar":"http://avatar.example.com"
   },
   "type":"text",
   "text":"Hello World!"
}
```

Post parameters

Name	Description	Validation
from	Unique Viber user ID of one of the account's admins. A list of admins and IDs is available in `get_account_info` response	Valid user ID of an account admin
type	Message type	required. Available message types: `text`, `picture`, `video`, `file`, `location`, `contact`, `sticker` and `url`
sender.name	The sender's name to display	optional. Max 28 characters
sender.avatar	The sender's avatar URL	optional. Avatar size should be no more than 100 kb. Recommended 720x720

Possible message types

The posted messages can be of all types supported by the `send_message` API: `text`, `picture`, `video`, `file`, `location`, `contact`, `sticker` and `url`.

The type of the message will be set by the `type` parameter and the content of message will be sent according to same parameters and logic as `send_message` requests (see send message section for more details.

Note: This method does not support keyboard attachment.

Callbacks

Each callback will contain a signature on the JSON passed to the callback. The signature is HMAC with SHA256 that will use the authentication token as the key and the JSON as the value. The result will be passed as HTTP Header `X-Viber-Content-Signature` so the receiver can determine the origin of the message.

Re-try logic

In case the webhook is offline Viber will re-try to deliver the callback several times for up to an hour until HTTP status code `200` is received.

Input

Key:

```
auth_token - 4453b6ac12345678-e02c5f12174805f9-daec9cbb5448c51f
```

Value:

```
{
   "event":"delivered",
   "timestamp":1457764197627,
   "message_token":491266184665523145,
   "user_id":"01234567890A="
}
```

Output

HTTP header

```
X-Viber-Content-Signature: 9d3941b33d45c165400d84dba9328ee0b687a5a18b347617091be0a56d
```

Subscribed

Before an account can send messages to a user, the user will need to subscribe to the account. Subscribing can take place in one of two ways:

1. User sends message to the account (both Public Accounts and bots) - when a user sends its first message to a account the user will be automatically subscribed to the account. Sending the first message will not trigger a subscribe callback, only a message callback (see receive message from user section).
2. Subscribed event is sent to the Public Account (Public Accounts only) - user clicks a subscribe button which triggers the subscribe callback as described below.

Note: A subscribe event will delete any `context` or `tracking_data` information related to the conversation. This means that if a user had a conversation with a service and then chose to unsubscribe and subscribe again, a new conversation will be started without any information related to the old conversation.

Callback data

```
{
   "event":"subscribed",
   "timestamp":1457764197627,
```

```
   "user":{
      "id":"01234567890A=",
      "name":"John McClane",
      "avatar":"http://avatar.example.com",
      "country":"UK",
      "language":"en",
      "api_version":1
   },
   "message_token":4912661846655238145
}
```

Callback parameters

Name	Description	Possible values
event	Callback type - which event triggered the callback	subscribed
timestamp	Time of the event that triggered the callback	Epoch time
user.id	Unique Viber user id	
user.name	User's Viber name	
user.avatar	URL of user's avatar	
user.country	User's 2 letter country code	ISO ALPHA-2 Code
user.language	User's phone language. Will be returned according to the device language	ISO 639-1
user.api_version	The maximal Viber version that is supported by all of the user's devices	
message_token	Unique ID of the message	
chat_hostname	Internal use	

Unsubscribed

The user will have the option to unsubscribe from the PA. This will trigger an unsubscribed callback.

Callback data

```
{
   "event":"unsubscribed",
   "timestamp":1457764197627,
   "user_id":"01234567890A=",
```

```
    "message_token":4912661846655238145
}
```

Callback parameters

Name	Description	Possible values
event	Callback type - which event triggered the callback	unsubscribed
timestamp	Time of the event that triggered the callback	Epoch time
user_id	Unique Viber user id	
message_token	Unique ID of the message	

Conversation started

Conversation started event fires when a user opens a conversation with the Public Account/ bot using the "message" button (found on the account's info screen) or using a deep link.

This event is **not** considered a subscribe event and doesn't allow the account to send messages to the user; however, it will allow sending one "welcome message" to the user. See sending a welcome message below for more information.

Once a `conversation_started` callback is received, the service will be able to respond with a JSON containing same parameters as a `send_message` request. The `receiver` parameter is **not** mandatory in this case.

Callback data

```
{
   "event":"conversation_started",
   "timestamp":1457764197627,
   "message_token":4912661846655238145,
   "type":"open",
   "context":"context information",
   "user":{
      "id":"01234567890A=",
      "name":"John McClane",
      "avatar":"http://avatar.example.com",
      "country":"UK",
      "language":"en",
      "api_version":1
   },
```

```
    "subscribed":false
}
```

Callback parameters

Name	Description	Possible values
event	Callback type - which event triggered the callback	`conversation_started`
timestamp	Time of the event that triggered the callback	Epoch time
message_token	Unique ID of the message	
type	The specific type of `conversation_started` event	`open` . Additional types may be added in the future
context	Any additional parameters added to the deep link used to access the conversation passed as a string. See deep link section for additional information	
user.id	Unique Viber user id	
user.name	User's Viber name	
user.avatar	URL of user's avatar	
user.country	User's 2 letter country code	ISO ALPHA-2 Code
user.language	User's phone language	
user.api_version	Max API version, matching the most updated user's device	
subscribed	indicated whether a user is already subscribed	`true` if subscribed and `false` otherwise

Sending a welcome message

The Viber API allows sending messages to users only after they subscribe to the account. However, Viber will allow the account to send one "welcome message" to a user as the user opens the conversation, before the user subscribes.

The welcome message will be sent as a response to a `conversation_started` callback, which will be received from Viber once the user opens the conversation with the account. To learn more about this event and when is it triggered see `Conversation started` in the callbacks section.

Welcome message flow

Sending a welcome message will be done according to the following flow:

1. User opens 1-on-1 conversation with account.

2. Viber server send `conversation_started` even to PA's webhook.
3. The account receives the `conversation_started` and responds with an HTTP response which includes the welcome message as the response body.

The welcome message will be a JSON constructed according to the `send_message` requests structure, but without the `receiver` parameter. An example welcome message would look like this:

```
{
   "sender":{
      "name":"John McClane",
      "avatar":"http://avatar.example.com"
   },
   "tracking_data":"tracking data",
   "type":"picture",
   "text":"Welcome to our bot!",
   "media":"http://www.images.com/img.jpg",
   "thumbnail":"http://www.images.com/thumb.jpg"
}
```

Note: The welcome message should be sent as the body of the HTTP response to the `conversation_started` event, and not to the `send_message` endpoint.

Message receipts callbacks

Viber offers message status updates for each message sent, allowing the account to be notified when the message was delivered to the user's device (`delivered status`) and when the conversation containing the message was opened (`seen status`).
If the message recipient is using their Viber account on multiple devices, each of the devices will return a delivered and a seen status. This means that several callbacks can be received for a single message.
If Viber is unable to deliver the message to the client it will try to deliver it for up to 14 days. If the message wasn't delivered within the 14 days it will not be delivered and no "delivered" or "seen" callbacks will be received for it.

Callback data

Delivered

```
{
   "event":"delivered",
   "timestamp":1457764197627,
   "message_token":4912661846655238145,
```

```
    "user_id":"01234567890A="
}
```

Seen

```
{
    "event":"seen",
    "timestamp":1457764197627,
    "message_token":4912661846655238145,
    "user_id":"01234567890A="
}
```

Callback parameters

Name	Description	Possible values
event	Callback type - which event triggered the callback	`delivered` , `seen`
timestamp	Time of the event that triggered the callback	Epoch time
message_token	Unique ID of the message	
user_id	Unique Viber user id	

Failed callback

The “failed” callback will be triggered if a message has reached the client but failed any of the client validations.

Since some of the message validations take place on the server while the others take place on the client, some messages may only fail after reaching the client. In such cases the flow will be as follows:

1. Message is sent.
2. Response with status 0 is received to indicate a successful request.
3. The message reaches the client and fails client validation.
4. “Failed” callback is sent to the webhook, containing the unique message token and a string explaining the failure.

Such message will not be displayed to the receiver and no “delivered” or “seen” callbacks will be returned for it.

Callback data

```
{
   "event":"failed",
   "timestamp":1457764197627,
   "message_token":4912661846655238145,
   "user_id":"01234567890A=",
   "desc":"failure description"
}
```

Callback parameters

Name	Description	Possible values
event	Callback type - which event triggered the callback	`failed`
timestamp	Time of the event that triggered the callback	Epoch time
message_token	Unique ID of the message	
user_id	Unique Viber user id	
desc	A string describing the failure	

Receive message from user

The Chat API allows the user to send messages to the PA. Excluding `file` type, all message types are supported for sending from Public Account/ bot to user and from user to Public Account/ bot. These include: `text`, `picture`, `video`, `contact`, `URL` and `location`. The following callback data describes the structure of messages sent from user to PA.

Callback data

```
{
   "event":"message",
   "timestamp":1457764197627,
   "message_token":4912661846655238145,
   "sender":{
      "id":"01234567890A=",
      "name":"John McClane",
      "avatar":"http://avatar.example.com",
      "country":"UK",
      "language":"en",
      "api_version":1
   },
   "message":{
```

```
        "type":"text",
        "text":"a message to the service",
        "media":"http://example.com",
        "location":{
            "lat":50.76891,
            "lon":6.11499
        },
        "tracking_data":"tracking data"
    }
}
```

Callback general parameters

Name	Description	Possible values
event	Callback type - which event triggered the callback	`message`
timestamp	Time of the event that triggered the callback	Epoch time
message_token	Unique ID of the message	
sender.id	Unique Viber user id of the message sender	
sender.name	Sender's Viber name	
sender.avatar	Sender's avatar URL	
sender.country	Sender's 2 letter country code	ISO ALPHA-2 Code
sender.language	Sender's phone language. Will be returned according to the device language	ISO 639-1
sender.api_version	The maximal Viber version that is supported by all of the user's devices	
message	Detailed in the chart below	

Callback message parameters

The callback message parameters depend on the type of message. For each message type, only the relevant parameters will be received.

Name	Description	Possible values
type	Message type	`text` , `picture` , `video` , `file` , `sticker` , `contact` , `url` and `location`
text	The message text	

Name	Description	Possible values
media	URL of the message media - can be `image`, `video`, `file` and `url`. Image/Video/File URLs will have a TTL of 1 hour	
location	Location coordinates	lat & lon within valid ranges
contact	`name` - contact's username, `phone_number` - contact's phone number and `avatar` as the avatar URL	`name` - max 128 characters. Only one `phone_number` per contact can be sent
tracking_data	Tracking data sent with the last message to the user	
media	URL of the message media - can be `image`, `video`, `file` and `url`. Image/Video/File URLs will have a TTL of 1 hour	
file_name	File name	Relevant for `file` type messages
file_size	File size in bytes	Relevant for `file` type messages
duration	Video length in seconds	Relevant for `video` type messages
sticker_id	Viber sticker id	Relevant for `sticker` type messages

Message status

Once a `200` OK response is received from the PA, the message status will change to `delivered` on the user's side. "Seen" status is not currently supported for messages sent from user to PA.

Error Codes

The following error codes will be returned with API responses. The `status` parameter will include the error code value while the `status_message` parameter will include the error name or a more specific string describing the error.

Value	Name	Description
`0`	`ok`	Success
`1`	`invalidUrl`	The webhook URL is not valid
`2`	`invalidAuthToken`	The authentication token is not valid
`3`	`badData`	There is an error in the request itself (missing comma, brackets, etc.)
`4`	`missingData`	Some mandatory data is missing

Value	Name	Description
5	`receiverNotRegistered`	The receiver is not registered to Viber
6	`receiverNotSubscribed`	The receiver is not subscribed to the account
7	`publicAccountBlocked`	The account is blocked
8	`publicAccountNotFound`	The account associated with the token is not a account.
9	`publicAccountSuspended`	The account is suspended
10	`webhookNotSet`	No webhook was set for the account
11	`receiverNoSuitableDevice`	The receiver is using a device or a Viber version that don't support accounts
12	`tooManyRequests`	Rate control breach
13	`apiVersionNotSupported`	Maximum supported account version by all user's devices is less than the `minApiVersion` in the message
14	`incompatibleWithVersion`	`minApiVersion` is not compatible to the message fields
15	`publicAccountNotAuthorized`	The account is not authorized
16	`inchatReplyMessageNotAllowed`	Inline message not allowed
17	`publicAccountIsNotInline`	The account is not inline
18	`noPublicChat`	Failed to `post to public account` . The bot is missing a Public Chat interface
19	`cannotSendBroadcast`	Cannot send broadcast message
20	`broadcastNotAllowed`	Attempt to send broadcast message from the bot
other	`General error`	General error

Note: Failing to send the authentication token in the header will result in an error with the `missing auth_token` message.

Forbidden File Formats

Extension	Format	Operating system(s)
ACTION	Automator Action	Mac OS
APK	Application	Android
APP	Executable	Mac OS
BAT	Batch File	Windows

Extension	Format	Operating system(s)
BIN	Binary Executable	Windows, Mac OS, Linux
CMD	Command Script	Windows
COM	Command File	Windows
COMMAND	Terminal Command	Mac OS
CPL	Control Panel Extension	Windows
CSH	C Shell Script	Mac OS, Linux
EXE	Executable	Windows
GADGET	Windows Gadget	Windows
INF1	Setup Information File	Windows
INS	Internet Communication Settings	Windows
INX	InstallShield Compiled Script	Windows
IPA	Application	iOS
ISU	InstallShield Uninstaller Script	Windows
JOB	Windows Task Scheduler Job File	Windows
JSE	JScript Encoded File	Windows
KSH	Unix Korn Shell Script	Linux
LNK	File Shortcut	Windows
MSC	Microsoft Common Console Document	Windows
MSI	Windows Installer Package	Windows
MSP	Windows Installer Patch	Windows
MST	Windows Installer Setup Transform File	Windows
OSX	Executable	Mac OS
OUT	Executable	Linux
PAF	Portable Application Installer File	Windows
PIF	Program Information File	Windows
PRG	Executable	GEM
PS1	Windows PowerShell Cmdlet	Windows
REG	Registry Data File	Windows
RGS	Registry Script	Windows

Extension	Format	Operating system(s)
RUN	Executable	Linux
SCT	Windows Scriptlet	Windows
SHB	Windows Document Shortcut	Windows
SHS	Shell Scrap Object	Windows
U3P	U3 Smart Application	Windows
VB	VBScript File	Windows
VBE	VBScript Encoded Script	Windows
VBS	VBScript File	Windows
VBSCRIPT	Visual Basic Script	Windows
WORKFLOW	Automator Workflow	Mac OS
WS	Windows Script	Windows
WSF	Windows Script	Windows