



WhatsApp

Agent Platform

Developer manual

Contents

Quick start.....	3
Step 1: Set up your agent and get your API token.....	3
Step 2: Receive messages.....	3
Step 3: Send a reply.....	3
Quick reference.....	4
Authentication.....	4
Participant identifiers.....	4
Stability.....	5
1. Send a message.....	5
POST /agent/v1/messages.....	5
2. Receive messages.....	9
GET /agent/v1/updates.....	9
3. Read receipts and typing indicator.....	16
POST /agent/v1/statuses.....	16
4. Media.....	18
POST /agent/v1/media.....	18
GET /agent/v1/media/<MEDIA_ID>.....	21
Download media.....	23
DELETE /agent/v1/media/<MEDIA_ID>.....	23
5. Errors.....	24
Send a message (POST /agent/v1/messages).....	25
Get updates (GET /agent/v1/updates).....	26
Statuses (POST /agent/v1/statuses).....	26
Media.....	27
Error reference.....	28
6. Rate limits.....	28

This document describes the API at <https://api.whatsapp.com/agent/v1>.

Quick start

Get your agent running and send your first message.

Step 1: Set up your agent and get your API token

Open WhatsApp, then **Settings > Agents > Create an agent**.

1. Set a display name and avatar
2. Open the agent's chat, then **Chat info > API key**
3. Copy the key — this is your API token, used to authenticate your requests
4. Store your API token securely; if you lose it, you must regenerate it

Note: If you uninstall the app, you must regenerate your API token.

Step 2: Receive messages

Receive messages by calling `GET /agent/v1/updates`: you will receive a response when messages arrive or when the timeout elapses.

```
curl 'https://api.whatsapp.com/agent/v1/updates?limit=50&timeout=15' \
-H 'Authorization: Bearer <ACCESS_TOKEN>'
```

If no messages arrive before the timeout elapses, you will receive an empty response.

Step 3: Send a reply

```
curl 'https://api.whatsapp.com/agent/v1/messages' \
-H 'Content-Type: application/json' \
-H 'Authorization: Bearer <ACCESS_TOKEN>' \
-d '{
  "messaging_product": "whatsapp",
  "to": "<RECIPIENT_ID>",
  "type": "text",
  "text": { "body": "Hello from my agent!" }
}'
```

For `<RECIPIENT_ID>`, use the `messages[] . from` value of an inbound message — one sent to your agent — exactly as you received it.

Quick reference

Item	Value
Base URL	<code>https://api.whatsapp.com/agent/v1</code>
Authentication	API token in the <code>Authorization</code> header
Rate limit	12 requests/minute for <code>POST /messages</code> , <code>POST /statuses</code> and each media method; 15 requests/minute for <code>GET /updates</code> .
Text message max length	4096 characters
Caption max length	1024 characters
Media size limits	5 MB image; 500 KB sticker; 16 MB video, audio, document, generic binary
Message types	Send: <code>text</code> , <code>image</code> , <code>audio</code> , <code>video</code> , <code>document</code> , <code>sticker</code> . Receive: those plus <code>reaction</code>
Update retention	30 days

Authentication

Attribute	Value
Type	API token
Header	<code>Authorization: Bearer <ACCESS_TOKEN></code>
Token format	Opaque string. Treat it as a secret and do not parse it.
Expiration	Rotation invalidates the token.

The API token identifies the agent.

If the `Authorization` header is absent or malformed, you will receive HTTP code `401` and `error.code 190`. If the token is present but not valid, you will receive HTTP code `400` and `error.code 100`.

Participant identifiers

A participant identifier is written as `<type>:<id>` — `user:<id>` for a WhatsApp user and `agent:<id>` for an agent. Treat the whole string as opaque: compare it in full, and never show it to a WhatsApp user.

Fields that identify a party use this form: `messages[].from`, `contacts[].wa_id` and `statuses[].recipient_id` on GET `/agent/v1/updates`; to on POST `/agent/v1/messages`; and `contacts[].wa_id` and `contacts[].input` on the POST `/agent/v1/messages` response.

`messages[].context.from` can contain either form: `user:<id>` when a WhatsApp user wrote the quoted message, `agent:<id>` when your agent did. Read the prefix to tell them apart.

to accepts `user:<id>` only. An `agent:<id>`, a bare number, or any other shape is rejected with 400 (code 131009).

Stability

An identifier refers to an account rather than to a person, and may change.

A WhatsApp user may receive a new identifier; changing phone number is a common case where this may occur. A user account that is deleted and later re-registered may also be assigned a new identifier.

To design for this:

- Treat an identifier as a conversation key, not as a primary key for a user record
- An unrecognized identifier is a new conversation, and a returning WhatsApp user whose identifier changed cannot be matched to their earlier one
- A WhatsApp user may no longer be reachable at an identifier you stored earlier. Use the `from` value of a recent inbound message rather than a long-lived stored one

1. Send a message

POST /agent/v1/messages

Send an outbound message from your agent to a WhatsApp user.

Request syntax

```
curl 'https://api.whatsapp.com/agent/v1/messages' \
-H 'Content-Type: application/json' \
-H 'Authorization: Bearer <ACCESS_TOKEN>' \
-d '
{
  "messaging_product": "whatsapp",
  "to": "<RECIPIENT_ID>",
  "type": "<MESSAGE_TYPE>",
  "<MESSAGE_TYPE>": { <MESSAGE_CONTENTS> }
}'
```

Header	Value
Authorization	Bearer <ACCESS_TOKEN>
Content-Type	application/json

Request parameters

Field	Type	Description	Example value
messaging_product	string	Required. Always "whatsapp".	whatsapp
to	string	Required. Recipient, as <code>user:<id></code> , and the agent's creator. Send the identifier from <code>GET /agent/v1/updates</code> back unchanged. See Participant identifiers for the accepted form.	user:50972923564215
type	string	Required. One of <code>text</code> , <code>image</code> , <code>audio</code> , <code>video</code> , <code>document</code> , <code>sticker</code> .	text
context	object	Optional. Reply context. <code>context.message_id</code> is the WhatsApp message id (wamid) of the message being quoted, and is required if you send the object at all.	<pre>{ "message_id": "wamid.HBgONTA5NzI5MjM1NjQyMTUVEgASGBQzQTVGOUMyRTFCN0Q0MDY4QUYzMQA=" }</pre>
text	object	Conditional. Required when <code>type = "text"</code> .	<pre>{ "body": "Hello from your agent" }</pre>
image	object	Conditional. Required when <code>type = "image"</code> .	<pre>{ "id": "6b1f2c4e-3a7d-4f19-9c02-8e5b7d0a1f34", "caption": "look at this" }</pre>
audio	object	Conditional. Required when <code>type = "audio"</code> .	<pre>{ "id": "0d9c7a41-52b8-4e6f-b3a1-7c94e2f6081d" }</pre>
video	object	Conditional. Required when <code>type = "video"</code> .	<pre>{ "id": "c47e91a0-8f35-4d2b-a6e9-1b03f5c8d724", "caption": "a clip" }</pre>
document	object	Conditional. Required when <code>type = "document"</code> .	<pre>{ "id": "9a3d5f70-1c6b-42e8-8f4a-0d27b91e5c36", "filename": "report.pdf" }</pre>

Field	Type	Description	Example value
<code>sticker</code>	object	Conditional. Required when <code>type = "sticker"</code> .	<code>{"id": "24f80b6c-7e13-49a5-bd0f-3c62a8194e7b"}</code>

Text message object

Field	Type	Description	Example value
<code>text.body</code>	string	Required. Message content. Max 4096 characters.	<code>Hello from your agent</code>
<code>text.preview_url</code>	boolean	Optional. Render a preview for the first URL in the body. The URL must begin with <code>http://</code> or <code>https://</code> . Default <code>false</code> .	<code>true</code>

Outbound media objects (image / audio / video / document / sticker)

Field	Type	Description	Example value
<code><type>.id</code>	string	Required. Opaque handle returned by <code>POST /agent/v1/media</code> .	<code>6b1f2c4e-3a7d-4f19-9c02-8e5b7d0a1f34</code>
<code><type>.caption</code>	string	Optional. Caption text, max 1024 characters. Available on <code>image</code> , <code>video</code> and <code>document</code> .	<code>look at this</code>
<code><type>.filename</code>	string	Optional. Original filename (<code>document</code> only).	<code>report.pdf</code>

Note: Upload the file with `POST /agent/v1/media` first and pass the returned `id`. You receive HTTP code 400 if it is omitted.

These 400 conditions are specific to sending a message:

- The payload object matching `type` is missing — `text` is required when `type=text`, and likewise for each type (for example, if `type=image` then `image` is required)
- A length cap is exceeded. If this is the case, you will receive code 100 with the field name as the message
- The media id is missing, unknown, or expired
- `type` is `reaction` — reactions are receive-only

Only the payload object matching `type` is sent. Fields in the other payload objects can still be validated, so you receive HTTP code `400` when one of them exceeds a length limit or omits a required field.

Response

Field	Type	Description	Example value
<code>messaging_product</code>	string	Always <code>"whatsapp"</code> .	<code>whatsapp</code>
<code>contacts[].input</code>	string	The <code>to</code> value echoed back exactly as you supplied it.	<code>user:50972923564215</code>
<code>contacts[].wa_id</code>	string	The WhatsApp user the message was routed to, as <code>user:<id></code> . Can be sent back as <code>to</code> as-is.	<code>user:50972923564215</code>
<code>messages[].id</code>	string	Server-assigned message id (wamid format).	<code>wamid.HBgONTA5NzI5MjM1NjQyMTUVEgARGBI5QTJDNEU2RjhCMEQxMzU3QUMA</code>

Each array contains one element.

Example request

```
curl 'https://api.whatsapp.com/agent/v1/messages' \
-H 'Content-Type: application/json' \
-H 'Authorization: Bearer <ACCESS_TOKEN>' \
-d '
{
  "messaging_product": "whatsapp",
  "to": "user:50972923564215",
  "type": "text",
  "text": { "body": "Hello! How can I help you?" }
}'
```

Example request: image message

```
curl 'https://api.whatsapp.com/agent/v1/messages' \
-H 'Content-Type: application/json' \
-H 'Authorization: Bearer <ACCESS_TOKEN>' \
-d '
{
  "messaging_product": "whatsapp",
  "to": "user:50972923564215",
```

```
"type": "image",
"image": {
  "id": "<MEDIA_ID>",
  "caption": "Check this out!"
}
}'
```

Example response

```
{
  "messaging_product": "whatsapp",
  "contacts": [
    { "input": "user:50972923564215", "wa_id": "user:50972923564215" }
  ],
  "messages": [
    { "id": "wamid.HBg..." }
  ]
}
```

Do not issue concurrent sends to the same recipient, as ordering between concurrent sends is not guaranteed.

Retrying a send

Whether to retry depends on the response to the first attempt:

- A 2xx means the message was sent. Record `messages[].id` and do not retry
- A 4xx means it was not sent. Fix the request or the token — retrying it unchanged fails the same way. The exception is 429, which you may retry after backing off
- A 503 with `error.code 131016` means the message was not accepted for delivery, so it was not sent. Send it again after backing off
- A 500, a connection reset, or your own read timeout leaves it unknown whether the message was sent. Decide in advance whether to retry, as a retry may send the message twice

To prevent an ordinary delay from leading to an unresolved state, configure the client read timeout significantly higher than the standard duration required for sending.

2. Receive messages

GET /agent/v1/updates

Long-poll endpoint for inbound messages and receipts. The connection stays open until an update arrives or the timeout elapses.

Request syntax

```
curl
'https://api.whatsapp.com/agent/v1/updates?offset=<OFFSET>&limit=<LIMIT>&timeout=<TIMEOUT>' \
-H 'Authorization: Bearer <ACCESS_TOKEN>'
```

Request parameters

Field	Type	Default	Range	Description	Example value
<code>offset</code>	integer	head at request time	≥ 0 , or absent	Optional. Pointer to read updates from. Use <code>next_offset</code> from the previous response. Omitting it starts from the head at the moment the request arrives, so a first poll with no <code>offset</code> skips the backlog and returns only what arrives while it is open; pass <code>offset=0</code> to read from the beginning. A negative value behaves as 0.	1287
<code>limit</code>	integer	50	≤ 100	Optional. Max entries to return. Out-of-range values are adjusted: 0 or negative becomes 50, above 100 becomes 100.	50
<code>timeout</code>	integer (sec)	15	0–25	Optional. How long to hold the connection open. Out-of-range values are adjusted: below 0 becomes 0, above 25 becomes 25.	15

Response

Field	Type	Description	Example value
<code>object</code>	string	Always <code>"whatsapp_agent_platform"</code> .	<code>whatsapp_agent_platform</code>
<code>entry[].id</code>	string	Your agent's numeric id. Exactly one entry is returned.	123456789
<code>entry[].changes[].field</code>	string	Always <code>"messages"</code> . Exactly one change is returned.	messages
<code>entry[].changes[].value.messaging_product</code>	string	Always <code>"whatsapp"</code> .	whatsapp

Field	Type	Description	Example value
<code>entry[].changes[].value.contacts[].wa_id</code>	string	The other party, as <code>user:<id></code> . Can be sent back as <code>to</code> unchanged.	<code>user:50972923564215</code>
<code>entry[].changes[].value.contacts[].profile</code>	object	Optional. One field, <code>name</code> — the WhatsApp user's profile name. The <code>profile</code> key appears only when the WhatsApp user has set a profile name, so check for the key rather than for an empty string. <code>contacts[]</code> contains at most one entry per WhatsApp user, and that entry may not include <code>profile</code> — accumulate names across polls rather than relying on a single response.	<code>{"name": "Alex"}</code>
<code>entry[].changes[].value.messages[]</code>	array	Inbound messages (see Message object). Always present; may be empty.	—
<code>entry[].changes[].value.statuses[]</code>	array	Delivery and read receipts for messages <i>your</i> agent sent (see Status object). Always present; may be empty. A poll can return only statuses, only messages, or both.	<code>[]</code>
<code>next_offset</code>	integer	Pointer to pass as <code>offset</code> in the next poll.	<code>1288</code>

Message object

Field	Type	Description	Example value
<code>from</code>	string	The sender, as <code>user:<id></code> . Can be sent back as <code>to</code> unchanged.	<code>user:50972923564215</code>
<code>id</code>	string	WhatsApp message id (wamid format).	<code>wamid.HBgONTA5NzI5MjM1NjQyMTUVEgASGBQzQTVGOUMyRTFCN0Q0MDY4QUYzMQA=</code>
<code>timestamp</code>	string	Unix timestamp in seconds, as a string.	<code>1736844652</code>

Field	Type	Description	Example value
type	string	text, image, audio, video, document, reaction or sticker. Exactly one same-named payload object is present.	text
context	object	Present only when the message quotes an earlier one. Two required strings: id, the wamid of the quoted message, and from, its author. The field is id here, while the send request uses message_id. Independent of type: both text and media messages can include it.	{ "id": "wamid.HBgONTA5NzI5MjM1NjQyMTUVEgARGBIzQjdFMTREMEMyOUE1Rjg2QjQA", "from": "agent:123456789" }
text.body	string	Text content (when type=text).	Hello agent
image	object	Contains id, mime_type, sha256 (Base64), and caption when one was sent.	{ "id": "998a7c17-60bf-4f40-a313-85f3723e2104", "mime_type": "image/jpeg" }
audio	object	Contains id, mime_type, sha256 (Base64), and voice, which is true on a voice note.	{ "id": "5f637c19-4070-4385-ba83-297373444771", "mime_type": "audio/ogg", "voice": true }
video	object	Contains id, mime_type, sha256 (Base64), and caption when one was sent.	{ "id": "451bd2ad-a7c8-42f7-8411-7577cf10c34e", "mime_type": "video/mp4" }
document	object	Contains id, mime_type, sha256 (Base64), and caption and filename when they were sent.	{ "id": "14ef9fff-8519-41e3-bfc6-1771e92c984e", "mime_type": "application/pdf" }
sticker	object	Contains id, mime_type, sha256 (Base64), and animated, which is false on stickers you receive.	{ "id": "2c3dde1d-72ac-471b-b38e-33a7e0f2678d", "mime_type": "image/webp" }
reaction	object	message_id — the wamid being reacted to — and emoji. An empty emoji means the reaction was removed. Receive-only.	{ "message_id": "wamid.HBgONTA5NzI5MjM1NjQyMTUVEgARGBIzQzA4QTZGNTFE0UU3NDJCQzYA", "emoji": "👍" }

Inbound media objects (image / audio / video / document / sticker)

Field	Type	Description	Example value
<code><type>.id</code>	string	Media handle. Pass it to <code>GET /agent/v1/media/<MEDIA_ID></code> to fetch the bytes.	<code>998a7c17-60bf-4f40-a313-85f3723e2104</code>
<code><type>.mime_type</code>	string	MIME type of the stored bytes.	<code>image/jpeg</code>
<code><type>.sha256</code>	string	SHA-256 digest of the stored bytes, Base64 encoded. <code>GET /agent/v1/media/<MEDIA_ID></code> returns the same digest as hex.	<code>qMVn+oGMEIV+My80Bie9mw1IYstCsS+jX78zoLErQak=</code>
<code><type>.caption</code>	string	Caption the WhatsApp user sent. Available on image, video and document.	<code>look at this</code>
<code>document.filename</code>	string	Filename the WhatsApp user sent.	<code>report.pdf</code>
<code>audio.voice</code>	boolean	<code>true</code> when the WhatsApp user recorded the audio in the chat rather than attaching a file. The key appears only on a voice note, so check for the key.	<code>true</code>
<code>sticker.animated</code>	boolean	<code>false</code> on stickers you receive.	<code>false</code>

Example: image message

```
{
  "from": "user:50972923564215",
  "id": "wamid.HBg...",
  "timestamp": "1736844652",
  "type": "image",
  "image": {
    "id": "998a7c17-60bf-4f40-a313-85f3723e2104",
    "mime_type": "image/jpeg",
    "sha256": "qMVn+oGMEIV+My80Bie9mw1IYstCsS+jX78zoLErQak=",
    "caption": "look at this"
  }
}
```

Example: voice note

```
{
  "from": "user:50972923564215",
  "id": "wamid.HBg...",
  "timestamp": "1736844652",
  "type": "audio",
  "audio": {
    "id": "5f637c19-4070-4385-ba83-297373444771",
    "mime_type": "audio/ogg",
    "sha256": "h2Es2hY3zgogBRD6H18H56MI1WEVYK73ScITHR/3wa4=",
    "voice": true
  }
}
```

Status object

Field	Type	Description	Example value
<code>id</code>	string	The wamid of the outbound message this receipt is about.	<code>wamid.HBgONTA5NzI5MjM1NjQyMTUVEgARGBIzQzA4QTZGNTFEOUU3NDJCQzYA</code>
<code>status</code>	string	<code>delivered</code> or <code>read</code> .	<code>read</code>
<code>recipient_id</code>	string	The WhatsApp user the message was sent to, as <code>user:<id></code> .	<code>user:50972923564215</code>
<code>timestamp</code>	string	Unix timestamp in seconds, as a string.	<code>1736844652</code>

Statuses and messages share one per-agent sequence, so a single poll can return both and one `next_offset` advances past them.

Store `next_offset` as a 64-bit signed integer, and pass it back as `offset` unchanged rather than computing a value of your own.

Example request

```
curl 'https://api.whatsapp.com/agent/v1/updates?offset=1286&limit=50&timeout=15' \
-H 'Authorization: Bearer <ACCESS_TOKEN>'
```

Example response

```
{
  "object": "whatsapp_agent_platform",
  "entry": [
    {
      "id": "123456789",
      "changes": [
        {
          "field": "messages",
          "value": {
            "messaging_product": "whatsapp",
            "contacts": [
              {
                "wa_id": "user:50972923564215",
                "profile": { "name": "Alex" }
              }
            ],
            "messages": [
              {
                "from": "user:50972923564215",
                "id": "wamid.HBg...",
                "timestamp": "1736844652",
                "type": "text",
                "text": { "body": "Hello agent" }
              }
            ],
            "statuses": []
          }
        }
      ]
    }
  ],
  "next_offset": 1287
}
```

Starting a new agent

Choose a starting point on the first poll, then use `next_offset` for later polls.

- **Only new traffic:** omit `offset` on the first request. It resolves to the head at the moment the request arrives. Do this *once*: an offset-less poll re-resolves the head, so a loop built on them can miss anything written between one response and the next request
- **Everything retained:** pass `offset=0`. This replays up to 30 days of backlog, so an agent that replies automatically will answer every message in it. Record which `messages[].id` values you have handled, or filter on `messages[].timestamp`, before you start replying

Retention

Buffered entries are kept for **30 days** from the moment they are stored, then expire. Polling does not consume them, so the same `offset` can be re-read for as long as the entries remain available. A message you have marked as read with `POST /agent/v1/statuses` may be deleted, and is then no longer returned by a poll at an earlier `offset`.

Empty responses

You receive HTTP code `204` when the timeout elapses with nothing to deliver. The body is empty — there is no `next_offset` on a `204`, so re-poll with the same `offset` value.

3. Read receipts and typing indicator

POST /agent/v1/statuses

Mark a message the WhatsApp user sent to your agent as read, and optionally show a typing indicator in the same call.

Request syntax

```
curl 'https://api.whatsapp.com/agent/v1/statuses' \
-H 'Content-Type: application/json' \
-H 'Authorization: Bearer <ACCESS_TOKEN>' \
-d '{
  "messaging_product": "whatsapp",
  "status": "read",
  "message_id": "<WAMID>"
}'
```

Request parameters

Field	Type	Description	Example value
<code>messaging_product</code>	string	Required. Always "whatsapp".	<code>whatsapp</code>
<code>status</code>	string	Required. Only "read" is settable by an agent.	<code>read</code>
<code>message_id</code>	string	Required. The wamid returned by <code>GET /agent/v1/updates</code> . Must be a message the WhatsApp user sent to this agent.	<code>wamid.HBgONTA5NzI5MjM1NjQyMTUVEgASGBQzQTVGOUMyRTFCN0Q0MDY4QUYzMQA=</code>

Field	Type	Description	Example value
typing_indicator	object	Optional. Send {"type": "text"} to show a typing indicator to the WhatsApp user. type is required inside the object and "text" is its only accepted value. Omit the field, or send null, to mark read without one.	{"type": "text"}

You receive HTTP code 503 when the read receipt or the typing indicator is not accepted.

The indicator disappears once you reply, or after **25 seconds**, whichever comes first. For a reply that takes longer than that, repeat the call to refresh the indicator. Show a typing indicator only when you are going to reply.

Response

```
{
  "success": true
}
```

A 200 response contains success: true.

Example request

```
curl 'https://api.whatsapp.com/agent/v1/statuses' \
-H 'Content-Type: application/json' \
-H 'Authorization: Bearer <ACCESS_TOKEN>' \
-d '
{
  "messaging_product": "whatsapp",
  "status": "read",
  "message_id": "wamid.HBg..."
}'
```

Example request: typing indicator

```
curl 'https://api.whatsapp.com/agent/v1/statuses' \
-H 'Content-Type: application/json' \
-H 'Authorization: Bearer <ACCESS_TOKEN>' \
-d '
{
  "typing_indicator": {
    "type": "text"
  }
}'
```

```
"messaging_product": "whatsapp",
"status": "read",
"message_id": "wamid.HBg...",
"typing_indicator": { "type": "text" }
}'
```

Example response

```
{
  "success": true
}
```

4. Media

POST /agent/v1/media

Upload media for use in outbound messages.

Request syntax

```
curl 'https://api.whatsapp.com/agent/v1/media' \
-H 'Authorization: Bearer <ACCESS_TOKEN>' \
-F 'messaging_product=whatsapp' \
-F 'type=<MIME_TYPE>' \
-F 'file=@<PATH_TO_FILE>'
```

Request parameters

Field	Type	Description	Example value
messaging_product	string	Required. Always "whatsapp".	whatsapp
file	binary	Required. File to upload.	@./invoice.pdf
type	string	Optional. MIME type of the file. For example: application/pdf. If you omit it, the file part's own Content-Type applies.	application/pdf

Response

```
{
  "id": "<MEDIA_ID>"
}
```

Pass that `id` as `<type>.id` on a subsequent `POST /agent/v1/messages`.

Example request

```
curl 'https://api.whatsapp.com/agent/v1/media' \
-H 'Authorization: Bearer <ACCESS_TOKEN>' \
-F 'messaging_product=whatsapp' \
-F 'type=application/pdf' \
-F 'file=@./invoice.pdf'
```

Accepted media types

The size limit is **5 MB** for an image, **500 KB** for a sticker, and **16 MB** for a video, audio file, document or generic binary file. You receive HTTP code `400` and `error.code 131053` for a file over the limit for its type, and `error_data.details` reports the actual size against the limit.

The upload accepts the MIME types listed below, and `application/octet-stream` for a generic binary file. You receive HTTP code `400` and `error.code 131053` for a declared MIME type outside that list.

The file must also satisfy the codec and profile constraints below.

Declare the MIME type in the `type` form field. If you omit it, the file part's own `Content-Type` is used; when the part supplies none, you receive HTTP code `400` and `error.code 131053`.

Image

Format	Extension	MIME type	Notes
JPEG	<code>.jpg</code> , <code>.jpeg</code>	<code>image/jpeg</code>	Preferred for photographs
PNG	<code>.png</code>	<code>image/png</code>	Preferred for graphics with transparency

Images must be **8-bit, RGB or RGBA**.

Keep the total pixel count at or below **25 megapixels** — 5000×5000 , or an equivalent area.

Video

Format	Extension	MIME type	Notes
MP4	.mp4	video/mp4	Preferred container
3GPP	.3gp	video/3gpp	—

Encode video with the **H.264** video codec and **AAC** audio, with one audio stream or none.

For the widest playback compatibility across WhatsApp clients, target the H.264 **Baseline** profile, or the **Main** profile without B-frames.

Write the `moov` atom ahead of `mdat` (`ffmpeg -movflags +faststart`) so playback can start before the file finishes downloading.

Audio

Format	Extension	MIME type	Notes
AAC	.aac	audio/aac	ADTS stream
MP4 audio	.m4a	audio/mp4	AAC in an MPEG-4 container
MP3	.mp3	audio/mpeg	—
AMR-NB	.amr	audio/amr	AMR-NB only
Opus in Ogg	.ogg	audio/ogg	Mono input only
Opus	.opus	audio/opus	Stored as <code>audio/ogg;</code> <code>codecs=opus</code>

Only Opus is supported in Ogg.

Audio is delivered as a plain attachment.

Document

Format	Extension	MIME type	Notes
PDF	.pdf	application/pdf	—
Plain text	.txt	text/plain	—
Word	.doc	application/msword	—
Word	.docx	application/vnd.openxmlformats-officedocument.wordprocessingml.document	—

Format	Extension	MIME type	Notes
Excel	.xls	application/vnd.ms-excel	—
Excel	.xlsx	application/vnd.openxmlformats-officedocument.spreadsheetml.sheet	—
PowerPoint	.ppt	application/vnd.ms-powerpoint	—
PowerPoint	.pptx	application/vnd.openxmlformats-officedocument.presentationml.presentation	—

Set `document.filename` to the file name, including its extension.

Sticker

Format	Extension	MIME type	Notes
WebP	.webp	image/webp	Static or animated when sending; stickers you receive are static

512 × 512 is the conventional size, and the canvas must be at most 4096 × 4096.

Upload a sticker like any other media and pass the returned id as `sticker.id`. Captions are not available on stickers.

Captions and filenames

Type	Caption	Filename
image	Yes, ≤1024 chars	—
video	Yes, ≤1024 chars	—
audio	—	—
document	Yes, ≤1024 chars	Yes
sticker	—	—

GET /agent/v1/media/<MEDIA_ID>

Fetch metadata for a media object — either one attached to an incoming message, or one you uploaded.

Request syntax

```
curl 'https://api.whatsapp.com/agent/v1/media/<MEDIA_ID>' \
-H 'Authorization: Bearer <ACCESS_TOKEN>'
```

Response

You receive JSON metadata.

Field	Type	Description	Example value
url	string	URL for the stored bytes. Fetch it with your API token in the <code>Authorization</code> header. When you fetch it and the media id is unknown or expired, you receive HTTP code 404 and <code>error.code</code> 100.	<code>https://lookaside.fbsbx.com/agent/v1/media/8f3a2b91-5c47-4e0d-9a6b-1d2e3f4a5b6c/content</code>
mime_type	string	MIME type of the stored bytes, if known.	<code>application/pdf</code>
sha256	string	SHA-256 hex digest of the stored bytes. The sha256 on an incoming media payload is Base64 of the same digest.	<code>a8c567fa818c10857e332f340627bd9b0d4862cb42b12fa35fbf33a0b12b41a9</code>
file_size	integer	Size of the stored bytes, in bytes.	<code>232145</code>
id	string	Echoes the requested media id.	<code>8f3a2b91-5c47-4e0d-9a6b-1d2e3f4a5b6c</code>
messaging_product	string	Always "whatsapp".	<code>whatsapp</code>

You receive HTTP code 400 and `error.code` 100 for an unknown id, and for an id belonging to a different agent.

Example response

```
{
  "url":
  "https://lookaside.fbsbx.com/agent/v1/media/8f3a2b91-5c47-4e0d-9a6b-1d2e3f4a5b6c/co
  ntent",
  "mime_type": "application/pdf",
  "sha256": "a8c567fa818c10857e332f340627bd9b0d4862cb42b12fa35fbf33a0b12b41a9",
  "file_size": 232145,
  "id": "8f3a2b91-5c47-4e0d-9a6b-1d2e3f4a5b6c",
```

```
"messaging_product": "whatsapp"
}
```

Download media

GET /agent/v1/media/<MEDIA_ID> returns a `url`. Fetch it with the same API token to download the stored bytes.

Request syntax

```
curl '<MEDIA_URL>' \
-H 'Authorization: Bearer <ACCESS_TOKEN>' \
-o '<FILE_NAME>'
```

Response

You receive HTTP code `200`, and the body contains the raw media bytes.

Example request

```
curl
'https://lookaside.fbsbx.com/agent/v1/media/8f3a2b91-5c47-4e0d-9a6b-1d2e3f4a5b6c/content' \
-H 'Authorization: Bearer <ACCESS_TOKEN>' \
-o 'invoice.pdf'
```

DELETE /agent/v1/media/<MEDIA_ID>

Delete a media object you uploaded, or one that was sent to you.

Request syntax

```
curl -X DELETE 'https://api.whatsapp.com/agent/v1/media/<MEDIA_ID>' \
-H 'Authorization: Bearer <ACCESS_TOKEN>'
```

Response

```
{
  "success": true
}
```

A 200 response contains `success: true`.

You receive HTTP code 400 and `error.code` 100 for an unknown id, and for an id belonging to a different agent.

Media expires 30 days after it is stored.

5. Errors

Error responses have this shape:

```
{
  "error": {
    "message": "(#131009) Missing or malformed required fields",
    "type": "OAuthException",
    "code": 131009,
    "error_data": {
      "messaging_product": "whatsapp",
      "details": "Invalid participant format; expected \"<type>:<int id>\", where
type = agent|user"
    },
    "fbtrace_id": "AW7bqWj4..."
  }
}
```

Field	Description
<code>error.message</code>	Human-readable summary. Most messages begin with the numeric code as <code>(#code)</code> ; a length-cap rejection contains only the field name. The format is not guaranteed — use <code>error.code</code> .
<code>error.type</code>	Always <code>OAuthException</code> , including for errors unrelated to authentication.
<code>error.code</code>	Numeric error code. Codes are listed under Error reference .
<code>error.error_data.details</code>	Human-readable description of the specific problem. The wording is not guaranteed. It is absent when no detail was supplied.

Field	Description
<code>error.error_data.messaging_product</code>	Always <code>whatsapp</code> when <code>error_data</code> is present.
<code>error.fbtrace_id</code>	Trace id. Include it in support requests.

Use `error.code` to identify the failure. The HTTP status indicates its broad class.

Send a message (POST /agent/v1/messages)

Code	Condition	What to do
200	The message was accepted. The response includes the wamid.	—
400	Schema violation — a required field is missing or a value is invalid.	Check that <code>messaging_product</code> , <code>to</code> , <code>type</code> and the payload object matching <code>type</code> are present.
400	The media id is missing, unknown, or expired.	Upload via <code>POST /agent/v1/media</code> first and pass the returned <code>id</code> .
400	<code>context.message_id</code> is not a wamid from this conversation.	Pass a wamid from <code>GET /agent/v1/updates</code> . Omit <code>context</code> to send without a quote.
400	Malformed <code>to</code> — not a <code><type>:<id></code> identifier, or an <code>agent:</code> identifier.	Read <code>error_data.details</code> for the specific problem. Send <code>user:<id></code> , ideally copied from an inbound message.
400	<code>type</code> is <code>reaction</code> .	Reactions are receive-only.
400	The media type named by <code>type</code> cannot be sent. <code>error.code</code> is 131009.	Read <code>error_data.details</code> . Send the content as <code>text</code> , or as another media type.
400	The media is larger than the limit for the <code>type</code> you are sending it as. <code>error.code</code> is 131009.	Send it as a type whose limit it fits, or upload a smaller file. Limits are listed under Accepted media types .
400	<code>text.body</code> is over 4096 characters, or a caption is over 1024.	Truncate before sending.
400	The API token is present but not valid.	Regenerate the token if it was rotated or revoked.
401	The <code>Authorization</code> header is absent or malformed.	Send the token as <code>Authorization: Bearer <ACCESS_TOKEN></code> .
403	The recipient is not the agent's creator.	An agent may only message its creator, and that account must have the agent API enabled. Verify <code>to</code> contains the creator's identifier in full, as <code>user:<id></code> .

Code	Condition	What to do
429	More than 12 sends per minute.	Back off exponentially.
500	Internal server error. The message may or may not have been sent.	Retry with exponential backoff – see Retrying a send . Include <code>error.fbtrace_id</code> in support requests.
503	The message was not accepted for delivery. <code>error.code</code> is 131016.	Send the message again after backing off – see Retrying a send .

Get updates (GET /agent/v1/updates)

Code	Condition	What to do
200	You receive one or more entries.	Process them and store <code>next_offset</code> for the next poll.
204	There are no entries at or after <code>offset</code> .	Re-poll with the same <code>offset</code> .
400	Malformed query parameters. Only a value that is not a plain integer is rejected.	Out-of-range values are adjusted instead: <code>limit</code> of 0 or less becomes 50 and above 100 becomes 100, <code>timeout</code> is adjusted into 0–25s, and a negative <code>offset</code> behaves as 0.
401	The <code>Authorization</code> header is absent or malformed. You receive HTTP code 400 for a token that is present but not valid.	Send the token as <code>Authorization: Bearer <ACCESS_TOKEN></code> .
409	A newer poll for this agent replaced this one.	Run one poll at a time for each agent.
429	More than 15 polls per minute.	Reuse the prior <code>offset</code> and back off exponentially.
500	Internal server error.	Reuse the prior <code>offset</code> and back off exponentially.

Statuses (POST /agent/v1/statuses)

Code	Condition	What to do
400	A field is missing or malformed. For a <code>message_id</code> that is not a wamid issued by this platform, <code>error_data.details</code> is Field "message_id" must be a wamid issued by this platform.	Fix the field named in <code>error_data.details</code> .
400	<code>typing_indicator</code> is present with a <code>type</code> other than "text", or with <code>type</code> missing.	Send <code>{"type": "text"}</code> , or omit the object.

Code	Condition	What to do
401	The <code>Authorization</code> header is absent or malformed. You receive HTTP code 400 for a token that is present but not valid.	Send the token as <code>Authorization: Bearer <ACCESS_TOKEN></code> .
403	The message was not sent by the agent's creator.	Mark as read only messages the agent's creator sent.
429	More than 12 requests per minute to this endpoint.	Back off exponentially.
500	Internal server error. The message may already be marked read.	Back off exponentially and send the request again.
503	The receipt, or the typing indicator, was not accepted. The two cases are not distinguishable, so the message may already be marked read.	Retry it.

Media

Code	Condition	What to do
200	GET returns metadata, POST returns <code>{"id": "<MEDIA_ID>"}</code> , and DELETE returns <code>{"success": true}</code> .	—
400	Invalid upload: <code>messaging_product</code> or <code>file</code> is missing. <code>error.code</code> is 131009.	Include the required form fields in the multipart request. <code>type</code> is optional.
401	The <code>Authorization</code> header is absent or malformed. You receive HTTP code 400 for a token that is present but not valid.	Send the token as <code>Authorization: Bearer <ACCESS_TOKEN></code> .
400	The id is unknown, expired, already deleted, or belongs to another agent. <code>error.code</code> is 100.	Re-request it from the sender, or upload the file again. You receive the same response for a malformed id.
404	The media id is unknown or expired on a fetch of the url from GET <code>/agent/v1/media/<MEDIA_ID></code> . <code>error.code</code> is 100.	Re-request the media from the sender, or upload the file again.
400	The upload exceeds the size limit for its type. <code>error.code</code> is 131053.	Re-encode or split the file. Limits are listed under Accepted media types .
400	The declared MIME type is not accepted. <code>error.code</code> is 131053.	Use one of the types listed under Accepted media types .
429	More than 12 requests per minute to a single media method.	Back off exponentially.
500	Internal server error.	Retry with exponential backoff.

Error reference

error.code	HTTP	Meaning
2	500	Internal server error.
100	400, 404	The API token is present but not valid, the media id is unknown, or you send <code>text.body</code> or a caption over its length cap. For an unknown media id, you receive HTTP code 400 from <code>GET</code> or <code>DELETE</code> <code>/agent/v1/media/<MEDIA_ID></code> , and HTTP code 404 when you fetch the url that <code>GET</code> returns and the media id is unknown or expired. For a length cap, <code>error.message</code> is the field name rather than a sentence.
190	401	The <code>Authorization</code> header is absent or malformed.
130429	429	Too many requests.
131005	403	The recipient is not the agent's creator on <code>POST /agent/v1/messages</code> , or the message being marked read was not sent by the agent's creator on <code>POST /agent/v1/statuses</code> .
131009	400	A required field is missing or malformed, the recipient is not a WhatsApp user, the media type cannot be sent, or the media id you pass to <code>POST /agent/v1/messages</code> is unknown or expired.
131016	503	Not accepted for delivery.
131053	400	Media rejected at upload — over the size limit for its type, or a MIME type that is not accepted (<code>POST /agent/v1/media</code> only).
1752041	409	A newer poll replaced this one.

6. Rate limits

Limit	Value	Scope
Outbound messages (<code>POST /agent/v1/messages</code>)	12/min	Per agent
Read receipts and typing (<code>POST /agent/v1/statuses</code>)	12/min	Per agent
Update polls (<code>GET /agent/v1/updates</code>)	15/min	Per agent
Media requests (<code>POST/GET/DELETE /agent/v1/media</code>)	12/min each	Per agent, per method

Each method has its own counter over a rolling 60-second window, per agent.