Large Account

WhatsApp Integration JSON API Guide

Doc Revision: 1.15 | 21 October 2026

This document is delivered subject to the following conditions and restrictions:

- This document contains proprietary information belonging to Cellact Ltd. Such information is supplied solely for the purpose of assisting explicitly and appropriately authorized users of Cellact products.
- No part of this document may be used for any other purpose, disclosed to any person or firm or reproduced by any means, electronic or mechanical, without the express prior written permission of Cellact Ltd.
- The software described in this document is furnished under a license. The Cellact software may be used or copied only by the terms of that agreement.
- The information contained in this document, specifications on which the document is based and the examples contained in it are all subject to change without notice.
- Corporate and individual names and data used in examples herein are fictitious unless otherwise noted. The text and the graphics included in this document are for instruction, illustration, and reference only.
- The names of other companies, product brands, and services mentioned in this document are trademarks or registered trademarks of their respective holders.

Copyright ©2026, Cellact Ltd. All rights reserved.

Contents

1 Before you start 7

1.1 Integration overview 7

1.2 Choose an account model 8

1.3 Information to prepare 8

1.4 Connection values supplied during onboarding 10

2 Preparing WhatsApp Business Platform 11

2.1 Create or identify the business account 11

2.2 System User and secure access 12

2.3 Connect Meta webhooks to Cellact 12

2.4 Create and approve templates 13

2.5 Recipient window and message choice 13

2.6 Summary: information to send to Cellact 14

3 Sending WhatsApp messages through LA 16

3.1 Endpoint and authentication 16

3.2 Request fields 17

3.3 First send with an approved template 18

3.4 Immediate response and correlation 18

3.5 Send text in an open window 19

3.6 Template components and buttons 19

3.7 Quoted replies and reactions 22

3.8 Use case: one-time password (OTP) 22

4 Uploading and sending media through LA 25

4.1 Media flow overview 25

4.2 Media service access 26

4.3 Initialize the upload 27

4.4 Transfer chunks in order 27

4.5 Complete and obtain the media ID 28

4.6 Send the media message 28

4.7 Supported media types 29
4.8 Python upload and send example 29
5 Receiving messages and downloading media 32
5.1 Configure the incoming endpoint 32
5.2 Callback security and retries 33
5.3 Incoming text event 34
5.4 Incoming media and download 35
5.5 Replies and reactions from the user 35
6 Delivery status and billing 36
6.1 Status endpoint and event structure 36
6.2 Safe event processing 37
6.3 Billing fields 37
7 Business information and service boundaries 38
7.1 Optional business lookup service 38
7.2 Cellact managed integration 39
8 Testing and going live 40
8.1 Acceptance test sequence 40
8.2 Retries and duplicate prevention 41
8.3 Troubleshooting 42
8.4 Production readiness 43
9 Appendix 44
9.1 Field and identifier reference 44
9.2 Delivery result codes (errorId) 44
9.3 References 47

List of Tables

Table 1 Integration paths 7
Table 2 Customer onboarding checklist 8
Table 3 Connection configuration 10
Table 4 Information to send to Cellact 14
Table 5 Information provided by Cellact 15
Table 6 WhatsApp send request fields 17
Table 7 errorCode values in the send response 19
Table 8 Supported media at a glance 26
Table 9 Standalone media message types 28
Table 10 Documented media limits 29
Table 11 Status callback retry schedule (when enabled) 33
Table 12 Incoming message fields 34
Table 13 Delivery status codes 36
Table 14 Billing metadata 37
Table 15 Optional business lookup routes 38
Table 16 LA integration responsibilities 39
Table 17 End-to-end acceptance tests 40
Table 18 Troubleshooting guide 42
Table 19 Identifier glossary 44
Table 20 Delivery result codes 44

List of Figures

Figure 1 Outbound delivery and asynchronous confirmation 7
Figure 2 Meta documentation screenshot showing account selection and API Setup 11
Figure 3 Meta documentation screenshot showing System User and asset assignment 12
Figure 4 Meta documentation screenshot explaining template review and approved status 13
Figure 5 Illustration of the OTP message on the recipient's phone 23
Figure 6 Media upload and send flow 25
Figure 7 Incoming message delivery to the customer MO endpoint 32

Purpose and Scope

This guide explains how to prepare a WhatsApp Business account, connect a customer application to Cellact Large Account (LA), send messages using JSON, transfer media through the LA Media service, and receive incoming messages and delivery reports. It covers setup, implementation, testing and production handover.

Audience

Customer developers, integration engineers and Meta account administrators. Business owners should complete the onboarding checklist before implementation. Examples use fictitious data; replace all sample identifiers, template names and URLs with your assigned values.

Meta screenshots show public documentation, not a live customer account. Documentation was checked on 30 September 2026; linked official pages provide current navigation and account requirements.

Service boundary

The customer sends WhatsApp messages to LA. Cellact operates the connection from LA to the WhatsApp Cloud API. Media and optional business-information calls use separately assigned LA service URLs. This guide does not require customers to deploy or administer any Cellact infrastructure.

1 Before you start

1.1 Integration overview

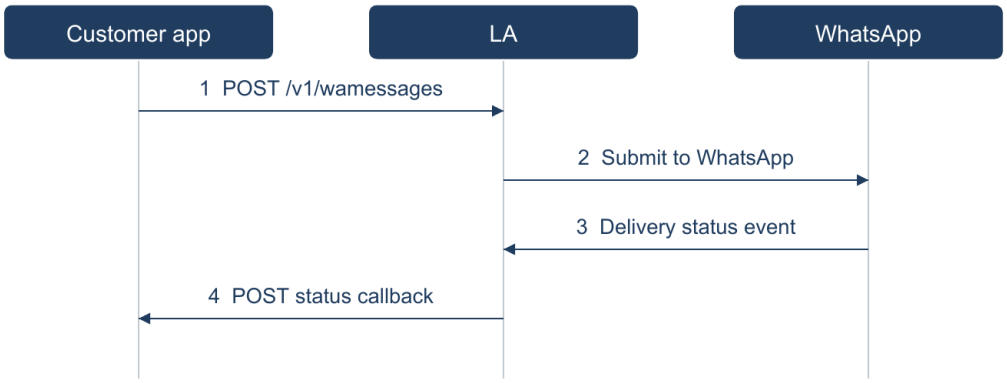
A complete integration has three customer-facing paths: outbound JSON requests to LA, media transfer through the LA service, and callbacks into your application. Meta account setup and template approval must be completed before the first production send.

Table 1 Integration paths

Path	Purpose	Credential
Customer → LA	Send text, templates, media messages, quoted replies and reactions.	LA API key
Customer → LA Media	Upload attachment bytes; obtain a WhatsApp media ID; download incoming files.	Per-business media token
LA → Customer	Deliver MO events and MT status notifications to configured HTTPS endpoints.	IP allowlist; unique URL; client certificate for status
LA ↔ Meta	Perform WhatsApp API calls and receive signed Meta events.	Meta credentials held by Cellact

MT means outbound message to a recipient. MO means an incoming message from a recipient. A successful HTTP send response acknowledges the request; asynchronous status events describe message progress. Store both your msg_id and the WhatsApp message ID (wamid).

Figure 1 Outbound delivery and asynchronous confirmation



1.2 Choose an account model

Choose one of the following models before you start. The choice determines whether you need your own Meta account and whether chapter 2 applies to you.

Your own Meta account (recommended). Messages are sent from your own WhatsApp Business account, business profile and number, while Cellact provides the sending infrastructure. Complete chapter 2, then continue with chapter 3.

Cellact-managed account (no Meta account needed). If you do not have a Meta business account, or prefer not to manage one, Cellact can allocate a WhatsApp number for you and/or register your number under the Cellact company Meta business account. Cellact then performs the Meta setup, and you send the content of each required template to Cellact Technical Support for creation and approval. Skip chapter 2 and continue with chapter 3; the LA sending, media and callback steps are the same. This model is provided under separate commercial terms.

1.3 Information to prepare

Send the following onboarding information to Cellact Technical Support. For multiple business numbers, provide a separate set of number-specific details and identify which templates and applications each number will use.

Table 2 Customer onboarding checklist

Item	What to provide
Account ownership	Choose your own Meta account (recommended) or the Cellact-managed model (section 1.2). For your own account, state whether it already exists and whether the number is connected to WhatsApp Business Platform.
Payment method	For your own account: confirm that a valid payment method is set for the WhatsApp Business account in Meta Billing Hub for the number you bring. Without it, Meta stops delivering messages once the free allowance is used, and charged messages are not delivered.
Business identifiers	Business phone number, WhatsApp Business Account ID (WABA ID), Business Phone Number ID and business name. The IDs are not the telephone number.
Secure credentials	System User details, System User Access Token and App Secret, delivered only through the secure channel agreed with Cellact.
Use case	Business purpose; send-only or two-way messaging; reply processing and customer-record matching requirements.
Templates	Exact template names, text, language codes, approval status, variables, media headers and required buttons.

Item	What to provide
Callbacks	Production and test MO/status endpoint URLs, HTTP method, payload format, required fields, authentication requirements, whether status callbacks should use a client certificate and retries, and technical contact.
Custom interfaces	Provide full API documentation, preferably OpenAPI/Swagger, when your receiver cannot accept the standard structure. Include requests, success/error responses and authentication.
Consent and opt-out	Describe how recipient permissions, consent and removal requests are managed. Confirm requirements with your legal adviser.
Capacity and timing	Expected volume, sending frequency, target launch date, peak load and special requirements.
Network and tokens	Public source IP addresses that will call LA and the media or business services, for allowlisting. Agree how the media/business token is created and transferred, and allow the Cellact callback source range 212.199.40.226–212.199.40.235 on your endpoints.

With the Cellact-managed model, the Meta-specific items in this checklist (business identifiers, System User credentials, payment method and template approval) are arranged with Cellact Technical Support instead of in your own Meta account.

With your own account, your team owns template creation and approval in your Meta account. In both models your team owns interpretation of replies, matching replies to customer records, and downstream business actions. A URL button leads to your website; your website handles identity, parameters and the resulting transaction. Custom interface development requires prior technical scoping and business approval.

1.4 Connection values supplied during onboarding

Table 3 Connection configuration

Setting	Value or action
LA_SEND_URL	https://la.cellactpro.com/v1/wamessages
LA_API_KEY	API key for your LA account, supplied by Cellact Technical Support. Used only for the LA send endpoint.
WA_SENDER	Your WhatsApp business number in international digits-only format, for example 972500000001. It is the telephone number of the Phone Number ID you register with Cellact. Before the first send, confirm with Cellact Technical Support that this number is associated with your account.
LA_USERNAME	Assigned LA username. Send it with WA_SENDER in every send, media and business-information request; the pair identifies your registered business.
MEDIA_BASE_URL	https://la-webhook-media-5filvqvaba-ez.a.run.app. Append /init, /chunk, /complete or /download.
MEDIA_TOKEN	Per-business Bearer token for the media and business-information services, registered by Cellact for your sender/username pair. If you are asked to generate it, use a strong random value of at least 32 bytes and transfer it only through the secure channel.
BUSINESS_BASE_URL	Business-information service base URL supplied by Cellact, if the optional service is enabled.
MO_URL / STATUS_URL	Your separately identified incoming-message and delivery-status endpoints; use HTTPS and arrange network access with Cellact.

Use only the LA service URLs provided during onboarding. The service URLs are environment-specific onboarding values. The LA API key, media token, Meta access token, App Secret and webhook verification token are different credentials and must not be interchanged.

2 Preparing WhatsApp Business Platform

This chapter applies when you use your own Meta account. If Cellact manages the Meta account for you (section 1.2), skip this chapter and continue with chapter 3.

2.1 Create or identify the business account

Sign in to the Meta App Dashboard with an authorized administrator account. Create an app using the Connect with customers through WhatsApp use case, or open your existing app. Select the appropriate business portfolio. In Customize use case > Connect on WhatsApp > Quickstart, choose Start using the API to open API Setup. Select or create the WhatsApp Business account and connect the intended phone number. [M1, M2]

Record the WABA ID and Phone Number ID for the production number. Meta may label the account as a Messaging account in newer screens. Match each ID to the correct business number; a test number and production number have different identifiers. Complete number verification, business profile and any account-specific Meta requirements before production. Add a valid payment method for the WhatsApp Business account in Meta Billing Hub; without one, Meta does not deliver charged messages, and service messages stop once the monthly free tier is used. [M5] Coordinate an already-registered number or a migration with Cellact before making changes.

Figure 2 Meta documentation screenshot showing account selection and API Setup



Step 2. Start using the API

1. Click the **Start using the API** button to set up the API by adding a phone number and sending your first message. You are redirected to the **API Setup** page.
2. In the **API Setup** section, connect your app to a Messaging account. This connection allows your app to access the WhatsApp Cloud API and send messages on behalf of your business. Select an existing Messaging account or create a new one:
   - **To use an existing account:** Select the Messaging account from the dropdown menu.
   - **To create a new account:** Click **Create a WhatsApp Business account** and follow the prompts to set up your business profile.
3. Once connected, you will see your Messaging account ID displayed in the API Setup panel.
   - Save this ID for use in API calls.

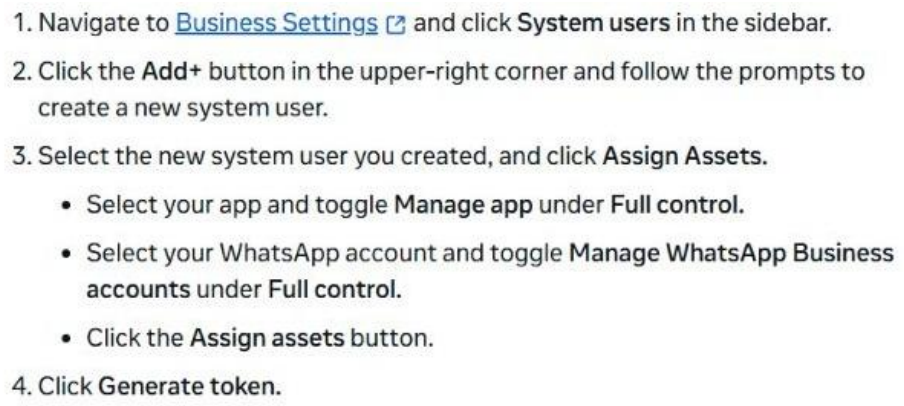
Screenshot source: Meta for Developers, WhatsApp Cloud API Get Started [M1], captured 30 September 2026. These are public documentation screens; menus and labels in your administrator console may differ. Source page: WhatsApp Cloud API Get Started.

2.2 System User and secure access

In Meta Business Settings, open System users, create or select the integration system user, and assign the app and WhatsApp account assets. Generate a system-user token for the intended app with whatsapp_business_messaging and whatsapp_business_management permissions. Add business_management only when your setup needs programmatic business-portfolio access. Follow the asset-access requirements shown by Meta. A short-lived API Setup test token is unsuitable for production. [M1, M2]

Record the System User identity and retrieve the App Secret from the app settings using the authorized administrator account. Transfer the access token and App Secret to Cellact through the agreed secure channel. Do not include secrets in ordinary email, screenshots, example payloads, browser-side JavaScript or application logs. Track token expiry and coordinate replacement with Cellact before it affects service.

Figure 3 Meta documentation screenshot showing System User and asset assignment


1. Navigate to Business Settings and click **System users** in the sidebar.
2. Click the **Add+** button in the upper-right corner and follow the prompts to create a new system user.
3. Select the new system user you created, and click **Assign Assets.**
   - Select your app and toggle **Manage app** under **Full control.**
   - Select your WhatsApp account and toggle **Manage WhatsApp Business accounts** under **Full control.**
   - Click the **Assign assets** button.
4. Click **Generate token.**

Screenshot source: Meta for Developers [M1], captured 30 September 2026. Apply the permissions needed for your integration; the screenshot includes Meta general-purpose setup guidance. Open Business Settings.

2.3 Connect Meta webhooks to Cellact

Cellact supplies the Meta callback URL and verification-token arrangement. Configure the WhatsApp webhook in the app Configuration area and complete Verify and Save with Cellact. The callback must reach the Cellact webhook service; your application MO_URL is the downstream destination, not a substitute for the Meta callback. Cellact coordinates the required messages subscription and verifies both incoming messages and delivery notifications. [M2]

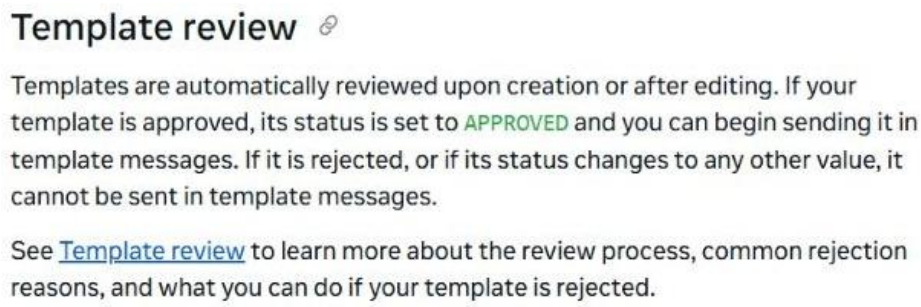
On the cloud side, Meta verifies the callback using hub.mode, hub.verify_token and hub.challenge. Incoming Meta POST requests are checked using X-Hub-Signature-256 and the registered App Secret. These checks apply to Meta → LA. They do not imply that the downstream customer callback carries the same Meta signature.

2.4 Create and approve templates

Open WhatsApp Manager > Message templates. Choose the correct account and create each required template with its name, category and language. Define the header, body, variables and buttons, provide sample variable values and sample media when required, and submit for review. Use the approved name and language exactly in API requests. Creation and approval remain your responsibility when you own the account. [M2, M3]

Maintain a template inventory: purpose, name, language, approval status, parameter order or names, header type, button index and URL design. Do not send a template that is pending, rejected, paused or disabled. Recheck status after editing or when delivery failures occur. [M3]

Figure 4 Meta documentation screenshot explaining template review and approved status



Screenshot source: Meta for Developers, Template fundamentals [M3], captured 30 September 2026. See Template review and Template fundamentals.

2.5 Recipient window and message choice

A message from a user opens a 24-hour customer-service window; each new user message resets that window. Send free-form text or media while it is open. Outside the window, or before the user has contacted you, send an approved template. Store the last incoming-message time per business number and recipient. Template delivery alone does not open a new customer-service window. [M1, M3, M4]

Meta free tier for service messages. Meta charges for service messages (free-form messages sent in an open customer-service window) per delivered message, and can provide a monthly free allowance of service messages for each business phone number. The allowance applies per number, not per WABA; service messages beyond it are charged. Template messages are always charged by category. [M5]

The free allowance, its size and its rules are part of Meta pricing policy, are subject to Meta terms and can change at any time; check the current Meta pricing page before relying on it. [M4, M5] Meta platform charges are separate from Cellact service charges. Use the billing data and your Cellact commercial agreement for accounting, and do not treat an open window as a guarantee that the complete Cellact service has no charge.

2.6 Summary: information to send to Cellact

When the steps in this chapter are complete, send the following information to Cellact Technical Support to start the service. Items marked Secret must be sent only through the secure channel agreed with Cellact Technical Support, never by ordinary email, chat, support tickets, screenshots or shared documents. If Cellact manages the Meta account for you (section 1.2), send only the template content and your callback details.

Table 4 Information to send to Cellact

Item	Where to find it	Sensitivity
Business phone number (WA_SENDER)	WhatsApp Manager > Phone numbers; international digits-only format	Standard
Business and display name	WhatsApp Manager > Phone numbers > Profile	Standard
WhatsApp Business Account ID (WABA ID)	App Dashboard > WhatsApp > API Setup	Standard
Phone Number ID	App Dashboard > WhatsApp > API Setup, for the production number	Standard
System User name and ID	Business Settings > System users	Standard
System User Access Token	Business Settings > System users > Generate token (section 2.2)	Secret (secure channel)
App Secret	App Dashboard > App settings > Basic	Secret (secure channel)
Payment method confirmation	Meta Billing Hub. Confirm that a method is set; never send card or bank details	Standard (confirmation only)
Template list	WhatsApp Manager > Message templates: name, language, category, variables, header type and buttons	Standard
MO_URL	Your incoming-message endpoint, including its random path segment (section 5.2)	Secret (secure channel)
Status endpoint and client certificate	Your status endpoint base URL, and whether you require a client certificate or retries (section 5.2)	Standard
Media and business service token	Only if you are asked to generate it (section 1.4)	Secret (secure channel)

Item	Where to find it	Sensitivity
Source IP addresses	Public addresses that will call LA and the media or business services	Standard
Technical contact	Name, email and telephone of the integration owner	Standard

Cellact Technical Support confirms the registration and returns the values below, which you need for the following chapters. Store the secret values in a protected configuration or secrets store, not in source code or logs.

Table 5 Information provided by Cellact

Item	Used for	Sensitivity
LA_API_KEY	Authorization for the LA send endpoint (chapter 3)	Secret (secure channel)
LA_USERNAME	LA account identity sent with WA_SENDER in every request	Standard
BUSINESS_BASE_URL	Optional business lookups (chapter 7)	Standard
MEDIA_TOKEN	Authorization for the media and business services, when Cellact issues it	Secret (secure channel)
Meta callback URL and verification token	Webhook configuration in your Meta app (section 2.3)	Secret (secure channel)
WA_SENDER confirmation	Confirmation that the business number is associated with your account	Standard

3 Sending WhatsApp messages through LA

3.1 Endpoint and authentication

```
POST https://la.cellactpro.com/v1/wamessages
Authorization: Bearer <LA_API_KEY>
Content-Type: application/json; charset=utf-8
Accept: application/json
```

WhatsApp uses /v1/wamessages. The SMS JSON API uses /v1/messages; the routes are different. Set cmd to sendwa and provide the WhatsApp content in meta. Do not use the legacy XML SMS/IVR endpoints for the JSON examples in this guide. Use one recipient per request, which is compatible with the current LA send contract. The SMS limit of 200 recipients does not establish a WhatsApp limit.

3.2 Request fields

Table 6 WhatsApp send request fields

Field	Use	Description
cmd	Required	Fixed value sendwa.
to_s	Required	Array containing one international recipient, for example ["+972500000002"].
sender	Required	Your registered WhatsApp business phone number. Use digits only in the canonical examples. It must match your Phone Number ID and be confirmed by Cellact Technical Support for your account.
username	Assigned	Use the LA username supplied during setup. The LA backend requires it; some LA configurations derive it from the API key.
webhook_url	Rec.	HTTPS status callback URL for this send. It is not the incoming-message (MO) destination.
msg_id	Rec.	Your unique correlation ID for this send. Persist before sending. It is not a documented idempotency key.
meta	Required	Message type and type-specific content.
meta.context. message_id	Cond.	wamid of the original message when quoting or reacting.

Rec. means recommended; Cond. means required for the relevant message type. The final field name is meta.context.message_id, written on two lines for display.

Use UTF-8 JSON and serialize with a JSON library. Normalize recipient numbers to E.164 with a leading +. The sender must match the business registered for the account. The examples in this guide use international recipient numbers and a digits-only sender.

3.3 First send with an approved template

Create and approve an order_update template in en_US with body variables {{1}} for the customer name and {{2}} for the order number. Save the following as request.json, substituting your own approved template and registered account details.

```
{
  "cmd": "sendwa",
  "to_s": ["+972500000002"],
  "sender": "972500000001",
  "username": "customer_account",
  "webhook_url": "https://customer.example/whatsapp/status",
  "msg_id": "order-12345-notice-001",
  "meta": {
    "type": "template",
    "template": {
      "name": "order_update",
      "language": {"code": "en_US"},
      "components": [
        {
          "type": "body",
          "parameters": [
            {"type": "text", "text": "Alex"},
            {"type": "text", "text": "12345"}
          ]
        }
      ]
    }
  }
}
```

```
curl --request POST \
  "https://la.cellactpro.com/v1/wamessages" \
  --header "Authorization: Bearer <LA_API_KEY>" \
  --header "Content-Type: application/json; charset=utf-8" \
  --header "Accept: application/json" \
  --data-binary @request.json
```

The cURL examples use POSIX shell line continuations. With Windows PowerShell, use curl.exe and place the command on one line or adapt the continuation characters. Never commit a real API key to a command file or source repository.

3.4 Immediate response and correlation

The LA response uses a result flag. Check both HTTP status and the JSON body. The following illustrates a successful LA envelope; it does not prove handset delivery.

```
{"result": true, "id": "gateway-request-id"}
```

Example of a failure response when the request body is not valid JSON:

```
{"result": false, "description": "invalid request", "errorCode": 12}
```

The send response can also include metadata.whatsapp_msg_id, request_id and msg_id. Your LA-facing implementation must not depend on those optional fields being returned synchronously. Obtain the outbound wamid from the status callback, retain it against your msg_id, and use it for replies and reactions.

When a POST /v1/wamessages request is rejected, the JSON response contains result set to false, a description string and a numeric errorCode. The following codes describe immediate send-request failures.

Table 7 errorCode values in the send response

errorCode	When	description
1	Authentication failed: invalid credentials, unauthorized IP or sender, or an inactive account.	invalid account
2	The user or API key is not recognized.	invalid account
10	The request structure or field values do not match the required JSON schema.	Validation error text
11	The source IP address is not allowed.	bad ip
12	The request body is not valid JSON.	invalid request

These errorCode values describe the immediate send response. Delivery callbacks use errorId for delivery status codes, listed in the appendix. Correct the credentials or request according to the error description before trying again.

3.5 Send text in an open window

After the recipient has sent a message and the customer-service window is open, use the same endpoint and envelope with a text payload.

```
{
  "cmd": "sendwa",
  "to_s": ["+972500000002"],
  "sender": "972500000001",
  "username": "customer_account",
  "webhook_url": "https://customer.example/whatsapp/status",
  "msg_id": "reply-12345-001",
  "meta": {
    "type": "text",
    "text": {"body": "Thank you. Your order is being checked."}
  }
}
```

3.6 Template components and buttons

For the next examples, replace meta in the send envelope. Preserve the other account and callback fields and use a new msg_id for each intended send. Omit components entirely for a template that has no parameterized components.

3.6.1 Text header and named body variables

Positional parameters follow the order in the approved component. A text header variable uses a header component with text parameters. Named body parameters use parameter_name; do not mix named and positional body parameters in one template.

```
{
  "type": "template",
  "template": {
    "name": "order_update_named",
    "language": {"code": "en_US"},
    "components": [
      {
        "type": "header",
        "parameters": [{"type": "text", "text": "Order update"}]
      },
      {
        "type": "body",
        "parameters": [
          {
            "type": "text",
            "parameter_name": "customer_name",
            "text": "Alex"
          },
          {"type": "text", "parameter_name": "order_id", "text": "12345"}
        ]
      }
    ]
  }
}
```

3.6.2 Template media header

For the documented template-header flow, supply a publicly fetchable HTTPS media URL. The header type must match the approved template. This URL is the actual send-time asset; media used as a sample during template approval is a separate step. Do not use a login-protected link, a localhost address or a private storage path.

```
{
  "type": "header",
  "parameters": [
    {"type": "image", "image": {"link": "https://cdn.example/banner.png"}}
  ]
}
```

The object above belongs inside meta.template.components. Use video or document with the corresponding object key when that is the approved header type. For a standalone attachment, use the media-ID flow in chapter 4.

3.6.3 Dynamic URL button and authentication code

The index is the zero-based button position in the approved template. For a dynamic URL button, send the value for its variable segment. Your website receives the navigation and performs customer matching and business processing; a URL click is not an MO text reply.

```
{
  "type": "button",
  "sub_type": "url",
  "index": "0",
  "parameters": [{"type": "text", "text": "order-12345"}]
}
```

For the documented authentication Copy Code pattern, repeat the same OTP in the body and the button parameter, using the approved authentication template. Use this pattern with LA; do not substitute a generic button schema for it. Section 3.8 shows the complete OTP flow.

```
{
  "type": "template",
  "template": {
    "name": "login_code",
    "language": {"code": "en_US"},
    "components": [
      {"type": "body", "parameters": [{"type": "text", "text": "4189"}]},
      {
        "type": "button",
        "sub_type": "url",
        "index": "0",
        "parameters": [{"type": "text", "text": "4189"}]
      }
    ]
  }
}
```

3.7 Quoted replies and reactions

Place context inside meta. Use the original incoming wamid or the outbound wamid learned from a status callback. Do not use msg_id, session_id or the LA request id as the WhatsApp target ID.

```
{
  "context": {"message_id": "wamid.ORIGINAL_MESSAGE"},
  "type": "text",
  "text": {"body": "Regarding your previous message..."}
}
```

The LA reaction format uses meta.context.message_id and meta.reaction.emoji. LA converts these into the WhatsApp format. The following is the meta object to place in the LA send envelope.

```
{
  "type": "reaction",
  "context": {"message_id": "wamid.ORIGINAL_MESSAGE"},
  "reaction": {"emoji": "👍"}
}
```

Use an empty emoji string to remove a reaction. Include reaction and quoted-reply cases in account acceptance testing, and apply the message-window restrictions for the chosen message type.

3.8 Use case: one-time password (OTP)

This example sends a login or transaction verification code with a WhatsApp authentication template and a Copy code button, using the pattern in section 3.6.3. Authentication templates are templates, so they can be sent at any time, inside or outside the customer-service window.

3.8.1 Create the authentication template

In WhatsApp Manager > Message templates, create a template in the Authentication category, for example login_code in en_US. Choose the Copy code option and, if required, enable the security recommendation and a code expiry time. Meta supplies the wording of authentication templates; you supply only the code when sending. Wait for APPROVED status before the first send. [M3]

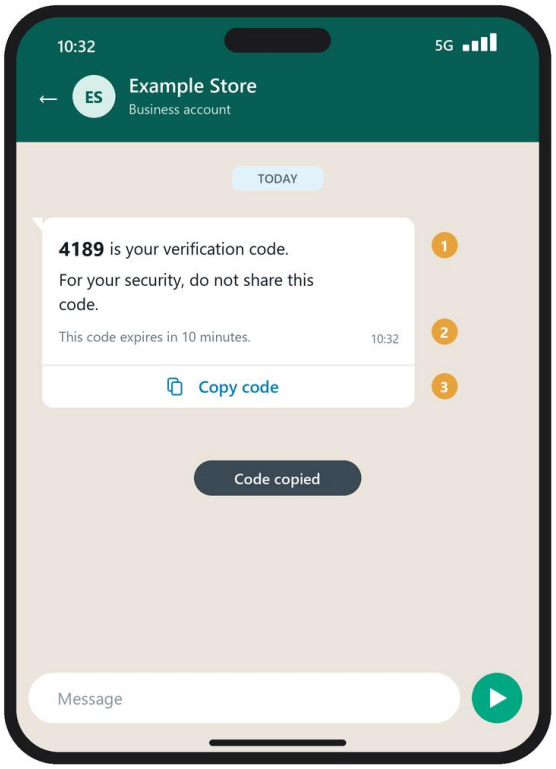
3.8.2 Send the code

Generate a random numeric code on your server and store only a hash of it, with its expiry time and your msg_id. Then send the template. The same code goes into the body parameter and the button parameter; Meta limits the code to 15 characters.

```
{
  "cmd": "sendwa",
  "to_s": ["+972500000002"],
  "sender": "972500000001",
  "username": "customer_account",
  "webhook_url": "https://customer.example/whatsapp/status/7d1f0c52",
  "msg_id": "otp-login-000123",
  "meta": {
    "type": "template",
    "template": {
```

```
      "name": "login_code",
      "language": {"code": "en_US"},
      "components": [
        {
          "type": "body",
          "parameters": [{"type": "text", "text": "4189"}]
        },
        {
          "type": "button",
          "sub_type": "url",
          "index": "0",
          "parameters": [{"type": "text", "text": "4189"}]
        }
      ]
    }
  }
}
```

Figure 5 Illustration of the OTP message on the recipient's phone



(1) Body with the code and the optional security recommendation, worded by Meta. (2) Optional expiry footer. (3) Copy code button; tapping it copies the code so the user can paste it into your app or website. The illustration is indicative; the exact appearance depends on the device, language and WhatsApp version.

3.8.3 Handle delivery and verification

Accept each code once and only until it expires. Track the status callback for the msg_id: mt_del confirms delivery to the handset. If you receive mt_rej or mt_nok, or no mt_del within your own time limit, offer another channel such as SMS through the Cellact SMS JSON API instead of resending repeatedly. Limit how many codes one number can request in a period. Never log the code or include it in support material. Authentication messages are reported in the authentication billing category; see section 6.3.

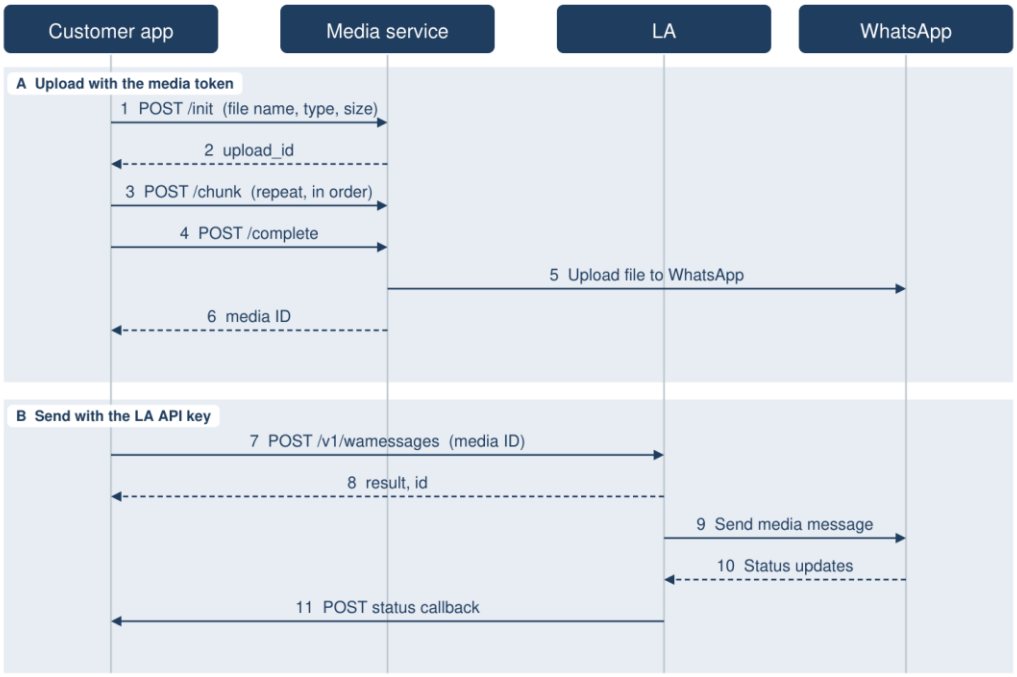
4 Uploading and sending media through LA

4.1 Media flow overview

Sending a media message has two stages. First upload the file to the Cellact media service, which stores it with WhatsApp and returns a WhatsApp media ID. Then send a message through LA that refers to that media ID. The two stages use different credentials: MEDIA_TOKEN for the upload and LA_API_KEY for the send.

- Upload to MEDIA_BASE_URL with the media token: /init declares the file, /chunk transfers the bytes in order, and /complete returns the media ID.
- Send to LA_SEND_URL with the LA API key: a message of type image, video, audio or document that contains the media ID.
- Track the message: LA reports its progress to the webhook_url of the send request (chapter 6).
- Alternative: when the file is already available on a public HTTPS URL, skip the upload and send the message with link instead of id.

Figure 6 Media upload and send flow



The following media can be sent. Section 4.7 lists the exact MIME types and limits.

Table 8 Supported media at a glance

Type	Formats	Maximum size	Caption
Image	JPEG, PNG	5 MB	Yes
Video	MP4, 3GPP	16 MB	Yes
Audio	AAC, MP4 audio, MPEG (MP3), AMR, OGG	16 MB	No
Document	PDF, Word, Excel, PowerPoint, plain text	100 MB	Yes, and a file name
Sticker	WebP	Enable with Cellact before use	No

4.2 Media service access

The media service base URL is https://la-webhook-media-5filvqvaba-ez.a.run.app. Cellact enables a per-business token for the registered sender/username pair. Use that token in Authorization: Bearer on every media call. The token is separate from the LA send API key. A business without a provisioned media token receives 403 from the media service.

```
MEDIA_BASE_URL = https://la-webhook-media-5filvqvaba-ez.a.run.app
Authorization: Bearer <MEDIA_TOKEN>

POST <MEDIA_BASE_URL>/init       application/json
POST <MEDIA_BASE_URL>/chunk      multipart/form-data
POST <MEDIA_BASE_URL>/complete   application/json
POST <MEDIA_BASE_URL>/download   application/json
```

The service stages the uploaded bytes and, on completion, uploads the file to WhatsApp Cloud API. The returned id is a WhatsApp media ID. It is not a storage object name, upload_id or public URL. Customers call the service with the media token only; no additional credentials are required.

4.3 Initialize the upload

```
{
  "sender": "972500000001",
  "username": "customer_account",
  "filename": "invoice.pdf",
  "content_type": "application/pdf",
  "total_size": 245331
}
```

POST this JSON to /init. total_size is the exact byte length. The file name is the original display name; content_type is the MIME type, not the file extension. On success:

```
{
  "result": true,
  "upload_id": "00000000-0000-4000-8000-000000000001",
  "chunk_size_hint": 8388608
}
```

4.4 Transfer chunks in order

POST each file chunk as multipart/form-data to /chunk using fields upload_id and chunk. Keep the Bearer header. Let your HTTP library create the multipart Content-Type boundary. Do not set application/json on this request. All chunks except the final chunk must be a multiple of 256 KiB (262,144 bytes). Eight MiB (8,388,608 bytes) is a suitable chunk size.

```
curl --request POST "<MEDIA_BASE_URL>/chunk" \
  --header "Authorization: Bearer <MEDIA_TOKEN>" \
  --form "upload_id=<UPLOAD_ID>" \
  --form "chunk=@part-000.bin;type=application/octet-stream"
```

In the current service, sender and username are resolved from the upload session for /chunk and /complete. Send upload_id and the same business token on these calls. Do not add a username field merely to create a session; /init establishes the association.

```
{
  "result": true,
  "bytes_received": 245331,
  "total_size": 245331,
  "complete": true
}
```

Send sequentially; complete is false while more bytes are expected and true when the declared total is received. Verify bytes_received and total_size. The upload session does not provide a documented client offset/resume API. If a chunk times out after transmission, do not blindly repeat it: the server may already have appended the bytes. Reconcile with support or start a fresh upload session under the agreed recovery procedure.

4.5 Complete and obtain the media ID

After the final chunk is accepted, POST the following JSON to /complete:

```
{"upload_id": "00000000-0000-4000-8000-000000000001"}
```

Successful completion returns the identifier used in the send payload:

```
{"result": true, "id": "1234567890123456"}
```

4.6 Send the media message

Send to LA_SEND_URL using the LA API key. The following complete example sends a PDF in an open customer-service window:

```
{
  "cmd": "sendwa",
  "to_s": ["+972500000002"],
  "sender": "972500000001",
  "username": "customer_account",
  "webhook_url": "https://customer.example/whatsapp/status",
  "msg_id": "invoice-12345-001",
  "meta": {
    "type": "document",
    "document": {
      "id": "1234567890123456",
      "filename": "invoice.pdf",
      "caption": "Your invoice for order 12345"
    }
  }
}
```

Table 9 Standalone media message types

Type	Payload under meta	Optional fields
image	type: image; image: {id: MEDIA_ID}	caption
video	type: video; video: {id: MEDIA_ID}	caption
audio	type: audio; audio: {id: MEDIA_ID}	No caption
document	type: document; document: {id: MEDIA_ID}	caption, filename

Raw bytes cannot be embedded in this send request. Upload first, then reference the returned media ID. Alternatively, replace id with link set to a publicly fetchable HTTPS URL; supply exactly one of id or link. Send from the business associated with the upload. Outside the customer-service window, use an approved media-header template instead of a free-form media message.

4.7 Supported media types

Table 10 Documented media limits

Category	MIME types	Limit
Image	image/jpeg; image/png	5 MB
Video	video/mp4; video/3gpp	16 MB
Audio	audio/aac; audio/mp4; audio/mpeg; audio/amr; audio/ogg	16 MB
Document	application/pdf; application/msword; application/vnd.ms-powerpoint; application/vnd.ms-excel; text/plain; corresponding Open XML MIME types below	100 MB

Open XML document MIME types are:

```
application/vnd.openxmlformats-officedocument.wordprocessingml.document
application/vnd.openxmlformats-officedocument.presentationml.presentation
application/vnd.openxmlformats-officedocument.spreadsheetml.sheet
```

These are the maximum sizes accepted by the service. Check the actual encoded file, use margin below the limit and handle server validation errors. A supported extension alone does not prove a supported codec. The service also recognizes stickers (image/webp); enable and validate that capability with Cellact before adding it to this customer workflow.

4.8 Python upload and send example

This example streams a local PDF through the three upload steps, obtains its media ID and sends it as a document message. Python 3 and the requests package are required. Set MEDIA_BASE_URL, MEDIA_TOKEN, WA_SENDER, LA_USERNAME, LA_SEND_URL, LA_API_KEY and STATUS_URL in your execution environment. It deliberately performs no automatic retries on chunk or send POSTs.

```
import os
from pathlib import Path
import requests

base_url = os.environ["MEDIA_BASE_URL"].rstrip("/")
headers = {"Authorization": "Bearer " +
           os.environ["MEDIA_TOKEN"]}

def checked(response):
    response.raise_for_status()
    data = response.json()
    if data.get("result") is not True:
        raise RuntimeError("Media operation rejected")
    return data

def upload_pdf(filename):
    path = Path(filename)
    total = path.stat().st_size
    if total == 0:
        raise ValueError("Empty file")
    init = checked(requests.post(
        base_url + "/init", headers=headers,
        json={"sender": os.environ["WA_SENDER"],
              "username": os.environ["LA_USERNAME"],
              "filename": path.name,
              "content_type": "application/pdf",
              "total_size": total}, timeout=(10, 120)))
    upload_id = init["upload_id"]
    unit = 256 * 1024
    hint = init.get("chunk_size_hint")
    size = (hint if isinstance(hint, int) and
            0 < hint <= 8 * 1024 * 1024 and
            hint % unit == 0 else 8 * 1024 * 1024)
```

Python upload and send example continued

```
    sent = 0
    with path.open("rb") as stream:
        while True:
            chunk = stream.read(size)
            if not chunk:
                break
            result = checked(requests.post(
                base_url + "/chunk", headers=headers,
                data={"upload_id": upload_id},
                files={"chunk": ("chunk.bin", chunk,
                         "application/octet-stream")},
                timeout=(10, 120)))
            sent += len(chunk)
            if result.get("bytes_received") != sent:
                raise RuntimeError("Byte count mismatch")
            if result.get("total_size") != total:
                raise RuntimeError("Total size mismatch")
            if result.get("complete") != (sent == total):
                raise RuntimeError("Completion mismatch")
    if sent != total:
        raise RuntimeError("File changed during upload")
    final = checked(requests.post(
        base_url + "/complete", headers=headers,
        json={"upload_id": upload_id}, timeout=(10, 120)))
    return final["id"]

def send_document(media_id, recipient, msg_id):
    envelope = {
        "cmd": "sendwa",
        "to_s": [recipient],
        "sender": os.environ["WA_SENDER"],
        "username": os.environ["LA_USERNAME"],
        "webhook_url": os.environ["STATUS_URL"],
        "msg_id": msg_id,
        "meta": {"type": "document",
                 "document": {"id": media_id,
                              "filename": "invoice.pdf"}}}
    response = requests.post(
        os.environ["LA_SEND_URL"], json=envelope,
        headers={"Authorization": "Bearer " +
                 os.environ["LA_API_KEY"]},
        timeout=(10, 60))
    response.raise_for_status()
    data = response.json()
    if data.get("result") is not True:
        raise RuntimeError("Send rejected: " + str(data))
    return data["id"]

media_id = upload_pdf("invoice.pdf")
la_id = send_document(media_id, "+972500000002",
                      "invoice-12345-001")
```

send_document posts the complete send envelope shown above and returns the LA id; store it with msg_id and wait for the status callback before treating the document as delivered. Keep the file unchanged throughout upload. Handle timeout and HTTP errors at your application boundary without logging Bearer tokens or raw customer documents. The timeouts in the sample are client choices, not server SLA values.

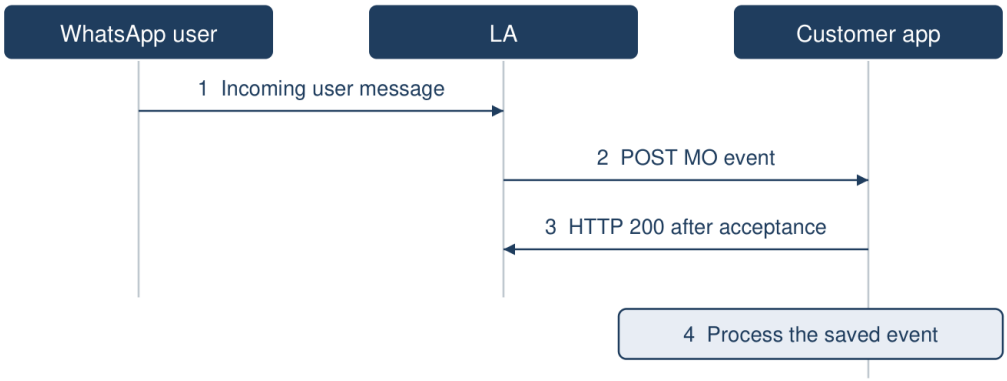
5 Receiving messages and downloading media

5.1 Configure the incoming endpoint

Provide Cellact with your MO endpoint during account setup. For the JSON flow in this guide, the server POSTs incoming events to that configured URL. This destination is configured separately from webhook_url on an outbound send. The general onboarding service can be assessed for HTTP GET/POST and XML/JSON, but alternative mappings must be agreed; they are not implied by these JSON examples.

Apply the callback security controls in section 5.2. MO callbacks are not retried, so keep this endpoint highly available. Accept valid events, persist or enqueue them durably, and respond HTTP 200 promptly. Perform media downloads and business processing after acknowledgment. A 200 response should mean you have accepted responsibility for processing the event.

Figure 7 Incoming message delivery to the customer MO endpoint



5.2 Callback security and retries

Incoming-message (MO) callbacks and delivery-status (MT) callbacks are protected and delivered differently. The source IP range applies to both; the other controls depend on the callback type.

Source IP allowlist (MO and status). Cellact sends all callbacks only from the fixed address range 212.199.40.226–212.199.40.235. Accept callback requests only from these addresses.

MO callbacks. MO_URL is a single URL configured for your account. Make it unique by including a long random path segment, and treat the full URL as a secret; Cellact Support can change it for you on request, for example to rotate that value. MO callbacks do not use a client certificate and are never retried, so keep the endpoint highly available and monitor it.

Status callbacks: client certificate. For status callbacks, Cellact can present a client-side TLS certificate (mutual TLS). Request the certificate details from Cellact Support and configure your status endpoint to require and validate it.

Status callbacks: unique URLs. You control webhook_url on every send, so you can give each request its own unguessable URL, for example https://customer.example/whatsapp/status/3f6c1e2a-8d4b-4c71-9a0e-5b2f7d913c48. Store the value with msg_id and reject status callbacks that do not match an outstanding send.

Status callbacks: retries. Status callbacks are not retried by default. Retries can be enabled for your account on request to Cellact Support. When enabled, a status callback is retried only when the request to your endpoint times out; other network errors, TLS failures and HTTP error responses are not retried. After the first attempt, up to four retries are made:

Table 11 Status callback retry schedule (when enabled)

Retry	Delay after previous attempt	Time after first attempt
1	2 minutes	2 minutes
2	10 minutes	12 minutes
3	1 hour	1 hour 12 minutes
4	2 hours	3 hours 12 minutes

Only timeouts are retried, so respond HTTP 200 quickly after durable acceptance and do not return an error status in the expectation of later redelivery. A request that timed out may already have been processed by your server, so a retry can deliver the same status event twice; apply the duplicate checks in section 6.2. An MO event, or a status event when retries are not enabled, that does not reach your endpoint is not redelivered.

The result codes reported in status callbacks are listed in Delivery result codes (errorId).

5.3 Incoming text event

```
{
  "event": "mo",
  "sender": "972500000002",
  "sender_name": "Alex Example",
  "to": "972500000001",
  "content": "Please check my order",
  "content_ref": "",
  "wamid": "wamid.INCOMING_MESSAGE",
  "id": "gateway-event-id",
  "session_id": "session-001",
  "when": "2026-09-30T10:32:00Z",
  "msg_type": "text"
}
```

Table 12 Incoming message fields

Field	Meaning
event / msg_type	event is mo. msg_type identifies the incoming kind; reaction has its own handling.
sender / sender_name	End-user (sender) phone number and reported WhatsApp display name. The display name is not an authenticated customer-record key.
to	Your business number receiving the message. Use it when separating multiple numbers.
content	Text or reaction emoji. It can be empty for an attachment.
content_ref	wamid of the quoted or reacted-to message, when present.
wamid / id	Store wamid as the WhatsApp identity. Retain id as a gateway fallback/event identity; do not assume it is a valid target for Meta replies.
session_id	Gateway conversation/session identifier.
when	Incoming-event timestamp; the documented example uses ISO 8601 UTC.
file	Present for an attachment; contains media_type, media_id, mime_type and filename when available.

Match the message to the correct business and customer in your application. A phone number match may need further verification for sensitive account actions. Interpretation of free-text answers, consent/opt-out handling and changes to business records remain your application responsibility.

5.4 Incoming media and download

An attachment event supplies a media reference, not the file bytes. For example, an MO event can contain the following file block:

```
{
  "file": {
    "media_type": "image",
    "media_id": "1234567890123456",
    "mime_type": "image/jpeg",
    "filename": "photo.jpg"
  }
}
```

POST to <MEDIA_BASE_URL>/download using the media Bearer token and this JSON:

```
{
  "sender": "972500000001",
  "username": "customer_account",
  "media_id": "1234567890123456",
  "filename": "photo.jpg"
}
```

On success, the response body is raw bytes, with Content-Type and Content-Disposition headers. Check the status before saving; errors use a JSON failure envelope. Stream the response to your controlled file store and validate the file before downstream use. Prefer media_id over a temporary media URL.

```
curl --request POST "<MEDIA_BASE_URL>/download" \
  --header "Authorization: Bearer <MEDIA_TOKEN>" \
  --header "Content-Type: application/json" \
  --data-binary @download.json \
  --fail-with-body --output photo.jpg
```

The cURL sample writes any error body into the output file too; inspect the exit code and HTTP result before treating it as an image. The download request also supports media_url as an alternative reference, but a stable media_id avoids expiry of the original URL.

5.5 Replies and reactions from the user

For an incoming quoted reply, content_ref points to the original wamid. For an incoming reaction, msg_type is reaction and content is the emoji, or an empty string when removed. Match content_ref to the stored message; do not treat the reaction as a new text answer.

```
{
  "event": "mo",
  "msg_type": "reaction",
  "sender": "972500000002",
  "to": "972500000001",
  "content_ref": "wamid.ORIGINAL_MESSAGE",
  "content": "👍"
}
```

6 Delivery status and billing

6.1 Status endpoint and event structure

Set webhook_url in each outbound request to your HTTPS status endpoint. Persist the outbound msg_id before transmitting so that a fast callback can be reconciled. Keep the LA response id, recipient and sender alongside that record.

```
{
  "id": "status-event-id",
  "sender": "972500000001",
  "to": "+972500000002",
  "msg_id": "order-12345-notice-001",
  "when": "20260930123434",
  "event": "mt_read",
  "errorId": "0",
  "description": "",
  "webhook_url": "https://customer.example/whatsapp/status",
  "billing": {
    "billable": true,
    "pricing_model": "PMP",
    "category": "utility",
    "type": "regular"
  },
  "session_id": "session-001",
  "metadata": {"whatsapp_msg_id": "wamid.OUTBOUND_MESSAGE"}
}
```

Table 13 Delivery status codes

Event	Meaning	Client action
mt_ok	Sent	Record send progress; this is not a delivery receipt.
mt_del	Delivered	Record delivery and trigger delivery-dependent actions once.
mt_read	Read	Record read status if reported.
mt_rej	Rejected	Retain errorId and description; correct the cause before resending.
mt_nok	Failed	Retain failure detail and apply an appropriate recovery decision.

metadata.whatsapp_msg_id is the WhatsApp message ID. msg_id is your correlation key; id is the gateway event identity. when in status examples is a compact yyyyMMddHHmmss string, unlike the ISO UTC example for MO. Status times are Israel local time (Asia/Jerusalem), including daylight saving time. Convert them with a time-zone database rather than a fixed offset, and keep the original string.

errorId and description carry the Cellact result code of a status event. For the meaning of each code, see Delivery result codes (errorId) in the appendix (section 9.2).

6.2 Safe event processing

Make both callback handlers tolerant of repeated or out-of-order events. Retain an event history and apply each business action once. Suggested keys include the business number plus wamid for messages, and the callback id plus event for status events. Do not move a delivered/read message back to sent when a delayed event arrives. Preserve error events separately for investigation.

Respond HTTP 200 after durable acceptance. Status callbacks are retried only when retries are enabled for your account, and MO callbacks are never retried; see section 5.2. Do not assume that every message will produce every status, or that absence of a read event means the user did not read it.

6.3 Billing fields

Table 14 Billing metadata

Field	Interpretation
billing.billable	Whether this reported send is marked billable. Check this boolean; presence of billing alone is insufficient.
billing.pricing_model	Applied pricing-model code, for example PMP.
billing.category	Reported category, for example utility, marketing, authentication or service.
billing.type	Reported pricing type: free_customer_service for a service message within the Meta monthly free tier; regular for a charged message.

Billing metadata is provided on MT confirmations in the documented gateway flow. MO events have no billing block. Reconcile per outbound message and recipient; do not count each sent/delivered/read callback as an additional billed message. Missing billing data means unknown, not automatically free.

Meta describes platform pricing on a per-delivered-message basis, with category and destination affecting the price. Service messages beyond the monthly free tier of each business phone number are charged (section 2.5). Consult the current official pricing pages and your Cellact agreement rather than hard-coding prices or using the older conversation-based terminology. [M4, M5]

7 Business information and service boundaries

7.1 Optional business lookup service

When enabled for your business, the LA business-information service provides read-only phone status and approved-template lookups. Use the assigned BUSINESS_BASE_URL and the same per-business token used for media. These are separate service routes, not suffixes on the LA /v1/wamessages URL. Access may also be subject to an IP allowlist.

Table 15 Optional business lookup routes

POST path	Request fields	Response
/phone-status	sender, username	result, phone_number_id, tier, quality_rating
/templates	sender, username	result and templates array of approved templates
/template	sender, username, name	result and template array of approved language variants

```
{"sender": "972500000001", "username": "customer_account"}
```

Use that request for /phone-status and /templates. A template detail request adds the approved name:

```
{
  "sender": "972500000001",
  "username": "customer_account",
  "name": "order_update"
}
```

The service returns 404 when the requested name has no active variant. A missing template in the approved-only list does not prove it was never created. Check WhatsApp Manager for pending, rejected, paused or disabled entries. tier and quality_rating can be null; do not treat null as unlimited capacity or a healthy rating.

7.2 Cellact managed integration

Table 16 LA integration responsibilities

Party	Responsibility	Interface
Customer application	Sends messages, uploads and downloads media, receives MO and status callbacks and processes replies.	LA_API_KEY, MEDIA_TOKEN, MO_URL, STATUS_URL
Customer Meta administrator	Owns the business account, number, templates, System User token and App Secret.	Meta Business Settings, WhatsApp Manager
Cellact provisioning	Registers the sender/username pair with the WABA ID, Phone Number ID, Meta credentials, callback mapping and media token.	Secure onboarding channel
Cellact LA platform	Accepts send requests and calls WhatsApp Cloud API on behalf of the business.	LA_SEND_URL
Cellact media and business services	Stages uploads, returns WhatsApp media IDs, downloads incoming media and reads phone status and approved templates.	MEDIA_BASE_URL, BUSINESS_BASE_URL
Cellact webhook service	Verifies Meta webhooks and signatures, then forwards MO and status events to your endpoints.	Meta callback URL supplied by Cellact

Registration associates your sender/username pair with the correct Meta assets. Cellact operates and secures the hosting, storage and credential management for these services. Internal Cellact service credentials are never issued to customers and are separate from the LA API key and the media token.

Template creation and approval are performed in Meta. The optional template lookup routes only read approved templates. Additional capabilities exposed by Meta, such as calling, groups or flows, are not automatically supported by this LA message interface.

8 Testing and going live

8.1 Acceptance test sequence

Table 17 End-to-end acceptance tests

Test	Expected result
1. Provisioning	LA key, sender/username and media/business tokens map to the intended business. Test and production callback URLs are recorded.
2. First template	An approved template reaches the test recipient. The returned LA id and status msg_id can be matched.
3. Delivery status	Status endpoint accepts events and stores wamid; repeated callbacks do not trigger duplicate actions.
4. Incoming text	Recipient reply arrives at MO_URL with sender, to, content and wamid.
5. Session reply	Text reply succeeds while the recipient window is open. Outside the window, the application selects an approved template.
6. Outbound media	PDF/image uploads complete with expected byte counts; the returned media ID produces the intended attachment.
7. Incoming media	MO file.media_id downloads to valid bytes with the expected Content-Type.
8. Templates and buttons	Variable order, language, header media and each URL/OTP button are verified on the receiving device.
9. Quote and reaction	A quoted reply points to the original message; reactions attach to the correct wamid and removal is handled.
10. Failure and recovery	Invalid token, unapproved template, unsupported media and a controlled callback failure produce understood outcomes.
11. Production handover	Volumes, monitoring, support contacts, consent handling and secure credential replacement are agreed. For your own account, a Meta payment method is in place.

8.2 Retries and duplicate prevention

A timeout may occur after acceptance. Reconcile uncertain sends against callbacks before resending: msg_id is not a documented idempotency key. Use bounded backoff only for recoverable failures under the agreed policy. Correct validation and credential errors first. Keep media chunks sequential and track confirmed byte counts; do not replay uncertain chunks or reuse an upload_id for another file.

8.3 Troubleshooting

Table 18 Troubleshooting guide

Symptom	Check and action
401 or authentication failure	Use LA_API_KEY for sends and MEDIA_TOKEN for media/business calls. Check Bearer formatting and environment selection.
403 on media or business	Confirm the business media token was provisioned and matches sender/username; check applicable business-route IP restrictions.
Template rejected	Check exact name, language, approved status, variable count/order and button/header definitions.
Free-form send rejected	Check the last incoming-message time. Use an approved template outside the service window.
Upload fails or returns 409	Check file size/type, ordered chunks, chunk-size multiples and upload session state. Do not replay uncertain chunks blindly.
No MO callback	Confirm the configured MO URL, Meta webhook subscription, business mapping, TLS/network access and receiver logs. Check the source IP allowlist. MO callbacks are never retried.
No status callback	Check webhook_url in the send request, reachable HTTPS endpoint and receiver acknowledgment. It must be the status destination. Check the IP allowlist and client-certificate settings; status callbacks are not retried by default.
Missing or wrong quote	Use meta.context.message_id and a real wamid. Check whether the original message belongs to this conversation.
HTTP 200 but no delivery	Check result in the immediate body and the later MT event/error detail. Acceptance is not handset delivery.
429 or temporary upstream error	Control sending rate and coordinate capacity with Cellact; preserve correlation data when investigating retry eligibility.

Failure envelopes can include result: false, errorCode, description and diagnostic metadata such as request_id. Log the safe error code, time, endpoint, msg_id, sender, recipient and request ID for support. Do not expose internal metadata to end users, and redact tokens and personal message content from support attachments.

8.4 Production readiness

Confirm production credentials, the Meta payment method (own account), templates, callbacks, capacity, monitoring, retention and opt-out ownership. Run a production smoke test with an authorized recipient. Retain the environment, business number, timestamp, HTTP result, LA id, msg_id, wamid and callback outcome as handover evidence. Agree custom interface changes with Cellact before deployment.

9 Appendix

9.1 Field and identifier reference

Table 19 Identifier glossary

Identifier	Used for
WABA ID	Identifies the WhatsApp business account that owns templates and phone assets.
Phone Number ID	Meta asset identifier for the business number. WA_SENDER is the telephone number that belongs to this ID; the ID itself is not sent as sender.
sender	Registered business number on outbound requests; user number on incoming MO events. Interpret by direction.
username	LA account identity used with sender to resolve the registered business.
msg_id	Customer-generated outbound correlation key.
wamid	WhatsApp message identity used for quote/reaction targets.
upload_id	Temporary media upload session, used for chunk and complete calls.
media ID	ID returned after upload completion or in an MO file block.
session_id	Gateway conversation/session identity.
MO / MT	Incoming user message / outgoing business message.

9.2 Delivery result codes (errorId)

Status (MT) callbacks report a Cellact result code in errorId, with the text in description; see Status endpoint and event structure (section 6.1) and Callback security and retries (section 5.2). A value of 0, as in the examples, means that no error was reported. The table lists the active Cellact result codes for WhatsApp messages.

Table 20 Delivery result codes

Code	Category	Description
1000	Delivery result	Message delivered to the subscriber device.

Code	Category	Description
5000	Delivery result	Message delivered to the operator.
7000	Delivery result	Subscriber responded to the MT message by clicking a link.
8000	Delivery result	WAP text message was opened by the subscriber.
9000	Delivery result	MO message delivered to the provider.
1002	Recipient	Subscriber is unavailable.
1003	Recipient	Message inbox full: the device has no space for new messages.
1005	Recipient	Incorrect subscriber number: disconnected or does not exist.
2006	Recipient	Destination is blocked to reverse-billed (RB) messages.
2010	Recipient	Subscriber is blocked from receiving messages.
2016	Recipient	Destination does not support the message type.
2060	Recipient	Kosher number block.
5007	Recipient	Recipient is temporarily unavailable or the device inbox is full.
9005	Recipient	Prepaid subscriber has insufficient balance for reverse-billed (RB) messages.
2013	Account and sender	Sender is invalid: the sender number does not exist or is disconnected.
2014	Account and sender	A subscriber on a different operator cannot be charged.
2050	Account and sender	Message was blocked by Guardian: the subscriber is blocked in the Guardian system.
2055	Account and sender	Account is blocked for sending messages abroad.
2057	Account and sender	Account is blocked for sending to landline numbers.
3001	Account and sender	Account message balance is exhausted.
3007	Account and sender	Night Block: the account cannot send messages during the configured hours.

Code	Category	Description
7200	Account and sender	Predefined SMS template name is missing.
2018	Message content	Unknown link: the link to the file is invalid.
2020	Message content	Filtered by the spam filter.
3002	Message content	Message contains invalid attributes.
5016	Message content	Message size exceeds the LA limit.
8018	Message content	MMS message rejected.
3004	Operator and system	MNP validator error: routing to the operator failed.
4002	Operator and system	Prepaid system did not recognize the message tariff.
5001	Operator and system	Operator error.
5014	Operator and system	System database connection error.
5028	Operator and system	Error sending an overseas message.
6001	Operator and system	Communication error with the operator.
7001	Operator and system	Message not delivered: general delivery failure.

Use errorId together with the event value: mt_rej and mt_nok carry the failure reason, while delivery results such as 1000 accompany successful events. Correct recipient, account and content errors before resending; retry operator and system errors only under your agreed policy.

9.3 References

Cellact reference: Cellact JSON API SMS Sending Guide (shared JSON envelope conventions). The LA WhatsApp send URL is https://la.cellactpro.com/v1/wamessages.

[M1] Meta WhatsApp Cloud API Get Started

[M2] Meta WhatsApp use case configuration

[M3] Meta Template fundamentals

[M4] WhatsApp Business Platform pricing

[M5] Meta Pricing on the WhatsApp Business Platform

Contacting Cellact

Cellact Technical Support

Telephone: +972-9-9704181 (09-9704181 within Israel)

Email: support@cellact.com

Coordinate account activation, secure credential transfer, LA service URLs, callback configuration and acceptance testing with the Cellact support team.

For a support case, provide the environment, affected business number, timestamp, msg_id and sanitized error details. Transfer sensitive information only through the secure channel agreed with support.