



MHI Gateway

MHI Gateway Quick Start Guide

Send and track an SMS in 5-10 minutes

Contract version 1.0.0-rc1
Document version 1.0
Publication date 28 July 2026
Classification Public integration documentation

MICRO HEALTH INITIATIVE

Document Control

Field	Value
Document title	MHI Gateway Quick Start Guide
Document identifier	MHI-SMS-QUICK-START-V1
Contract version	1.0.0-rc1
Document version	1.0
Status	Milestone 4
Audience	Developers completing their first SMS API integration
Authoritative specification	docs/api/openapi.yaml
Companion API reference	API Reference
Companion integration guide	Developer Integration Guide
Webhook implementation guide	Webhook Integration Guide

This guide provides the shortest contract-compliant path from credentials to a verified SMS integration. For field constraints and complete response matrices, use the [API Reference](#).

Revision History

Version	Date	Status	Description
1.0	28 July 2026	Initial	Official quick start for contract 1.0.0-rc1

Table of Contents

Document Control	2
Revision History	2
Table of Contents	3
1 Introduction	4
2 Prerequisites	5
3 Base URL	6
4 Authentication	7
5 Your First Request	8
6 Sending Your First SMS	9
6.1 cURL	10
6.2 PHP	11
6.3 Node.js	12
6.4 Python	13
6.5 Expected response	14
7 Reading Message Status	15
8 Configuring a Webhook	16
9 Common Errors	17
10 Verify Your Integration	18
11 Next Steps	19

1 Introduction

This guide walks through one complete integration path:

1. obtain a tenant-scoped API credential;
2. submit one SMS message;
3. save its public `message_id`;
4. retrieve its normalized status;
5. configure one signed delivery webhook;
6. confirm that identifiers and terminal outcomes reach your application.

The examples use placeholders and the frozen SMS API v1 contract. Replace placeholders only in private, environment-specific configuration. Never commit credentials, webhook secrets, or real message content.

2 Prerequisites

Before starting, obtain the following from the gateway operator:

Item	Example placeholder	Purpose
Environment host	https://<API_HOST>	Identifies the target environment
API credential	<API_TOKEN>	Authenticates one tenant and application
Test recipient	+255712345678	Receives the test SMS in E.164 format
Approved sender ID, if used	MHI	Identifies the sender

You also need:

- an HTTPS client such as cURL, PHP 8 with cURL, Node.js 18+, or Python 3.10+;
- a unique idempotency key for the submission;
- an HTTPS webhook URL if you will complete the webhook step;
- a cryptographically random webhook secret of at least 32 bytes.

The gateway operator controls credential issuance, expiry, and revocation. Store the supplied credential in a secret manager and inject it at runtime.

3 Base URL

The API prefix is:

```
/api/v1
```

Combine it with the environment host supplied by the gateway operator:

```
https://<API_HOST>/api/v1
```

The examples set this complete value as `BASE_URL`. Keep staging and production hosts, credentials, idempotency keys, and webhook secrets separate.

4 Authentication

Send the API credential as an HTTP Bearer token:

```
Authorization: Bearer <API_TOKEN>
```

The implemented credential form is:

```
mhi_<environment>_<12-character-prefix>.<48-character-secret>
```

Also send `Accept: application/json`. Requests with a JSON body require `Content-Type: application/json`.

A missing or invalid credential returns HTTP 401 with `AUTHENTICATION_FAILED`. Do not place the credential in a URL, request body, source file, or log.

5 Your First Request

Use the safe configuration-read operation to verify the host, TLS connection, and credential without sending a message:

```
curl --request GET \
  --url "https://<API_HOST>/api/v1/webhook" \
  --header "Authorization: Bearer <API_TOKEN>" \
  --header "Accept: application/json" \
  --header "X-Correlation-ID: quickstart-connectivity-1"
```

An authenticated request returns either 200 with the current webhook configuration or 404 with MESSAGE_NOT_FOUND when no configuration exists. Both outcomes confirm that the request reached the authenticated API surface.

Every HTTP response carries:

- X-Request-ID, unique to the current HTTP request; and
- X-Correlation-ID, the valid client-supplied value or a gateway-generated

value for the current request.

Capture both headers on successful and error responses.

6 Sending Your First SMS

Submit one SMS with POST /api/v1/messages. recipient, message, and sender_id are required. client_reference is optional.

The Idempotency-Key header is mandatory for this operation.

6.1 cURL

```
curl --request POST \
  --url "https://<API_HOST>/api/v1/messages" \
  --header "Authorization: Bearer <API_TOKEN>" \
  --header "Accept: application/json" \
  --header "Content-Type: application/json" \
  --header "Idempotency-Key: quickstart-message-0001" \
  --header "X-Correlation-ID: quickstart-request-0001" \
  --data '{
    "recipient": "+255712345678",
    "message": "Your verification code is 123456.",
    "sender_id": "MHI",
    "client_reference": "quickstart-0001"
  }'
```

6.2 PHP

```
<?php

$baseUrl = 'https://<API_HOST>/api/v1';
$token = '<API_TOKEN>';
$body = json_encode([
    'recipient' => '+255712345678',
    'message' => 'Your verification code is 123456.',
    'sender_id' => 'MHI',
    'client_reference' => 'quickstart-0001',
], JSON_THROW_ON_ERROR);

$handle = curl_init($baseUrl . '/messages');
curl_setopt_array($handle, [
    CURLOPT_POST => true,
    CURLOPT_POSTFIELDS => $body,
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_HEADER => true,
    CURLOPT_HTTPHEADER => [
        'Authorization: Bearer ' . $token,
        'Accept: application/json',
        'Content-Type: application/json',
        'Idempotency-Key: quickstart-message-0001',
        'X-Correlation-ID: quickstart-request-0001',
    ],
]);
$response = curl_exec($handle);
if ($response === false) {
    throw new RuntimeException('SMS request failed.');
}
echo $response;
curl_close($handle);
```

6.3 Node.js

```
const baseUrl = 'https://<API_HOST>/api/v1';

const response = await fetch(`${baseUrl}/messages`, {
  method: 'POST',
  headers: {
    Authorization: 'Bearer <API_TOKEN>',
    Accept: 'application/json',
    'Content-Type': 'application/json',
    'Idempotency-Key': 'quickstart-message-0001',
    'X-Correlation-ID': 'quickstart-request-0001'
  },
  body: JSON.stringify({
    recipient: '+255712345678',
    message: 'Your verification code is 123456.',
    sender_id: 'MHI',
    client_reference: 'quickstart-0001'
  })
});

console.log(response.status, Object.fromEntries(response.headers), await response.json());
```

6.4 Python

```
import requests

base_url = "https://<API_HOST>/api/v1"
response = requests.post(
    f"{base_url}/messages",
    headers={
        "Authorization": "Bearer <API_TOKEN>",
        "Accept": "application/json",
        "Content-Type": "application/json",
        "Idempotency-Key": "quickstart-message-0001",
        "X-Correlation-ID": "quickstart-request-0001",
    },
    json={
        "recipient": "+255712345678",
        "message": "Your verification code is 123456.",
        "sender_id": "MHI",
        "client_reference": "quickstart-0001",
    },
    timeout=30,
)
print(response.status_code, dict(response.headers), response.json())
```

6.5 Expected response

The gateway returns HTTP 202:

```
HTTP/1.1 202 Accepted
Content-Type: application/json
X-Request-ID: 01JREQUEST000000000000000000
X-Correlation-ID: quickstart-request-0001

{
  "message_id": "01JEXAMPLE0000000000000000",
  "status": "queued",
  "client_reference": "quickstart-0001",
  "created_at": "2026-07-19T12:30:40.123456Z",
  "correlation_id": "quickstart-request-0001"
}
```

Persist `message_id` and `client_reference`. The body `correlation_id` is the original submission correlation identifier and is retained if the same idempotency key and canonical request are replayed. Response headers always describe the current HTTP request.

7 Reading Message Status

Retrieve the compact status resource with:

```
curl --request GET \
  --url "https://<API_HOST>/api/v1/messages/01JEXAMPLE00000000000000/status" \
  --header "Authorization: Bearer <API_TOKEN>" \
  --header "Accept: application/json" \
  --header "X-Correlation-ID: quickstart-status-0001"
```

An HTTP 200 response has this shape:

```
{
  "message_id": "01JEXAMPLE00000000000000",
  "status": "delivered",
  "occurred_at": "2026-07-19T12:30:44.987654Z",
  "client_reference": "quickstart-0001",
  "correlation_id": "quickstart-request-0001"
}
```

The public statuses are queued, submitted, delivered, failed, expired, rejected, and unknown. The terminal statuses are delivered, failed, expired, and rejected.

For the full message representation, use GET /api/v1/messages/{message_id}. The full and compact status resources are distinct operations; see the [API Reference](#).

8 Configuring a Webhook

Configure the tenant's single delivery webhook with PUT `/api/v1/webhook`:

```
curl --request PUT \
  --url "https://<API_HOST>/api/v1/webhook" \
  --header "Authorization: Bearer <API_TOKEN>" \
  --header "Accept: application/json" \
  --header "Content-Type: application/json" \
  --header "X-Correlation-ID: quickstart-webhook-0001" \
  --data '{
    "url": "https://example.com/webhooks/mhi-sms",
    "secret": "<AT_LEAST_32_RANDOM_BYTES>",
    "enabled": true
  }'
```

The HTTP 200 response returns the safe configuration and never returns the secret:

```
{
  "url": "https://example.com/webhooks/mhi-sms",
  "enabled": true,
  "created_at": "2026-07-19T12:20:00.123456Z",
  "updated_at": "2026-07-19T12:20:00.123456Z"
}
```

When a message first reaches a terminal status, the gateway sends a signed POST containing `message_id`, `status`, `timestamp`, and the optional contract fields that apply. Verify `X-Webhook-Timestamp` and `X-Webhook-Signature` against the exact raw body before parsing JSON. Return any 2xx only after a durable handoff.

Signature verification, replay protection, secret rotation, and receiver examples are in the [Webhook Integration Guide](#).

9 Common Errors

Every error uses the standard JSON error envelope and includes a `correlation_id`. Capture the two response identifiers before troubleshooting.

HTTP	Code	First action
401	AUTHENTICATION_FAILED	Confirm the Bearer credential and environment
404	MESSAGE_NOT_FOUND	Confirm the public ID and authenticated scope
409	IDEMPOTENCY_CONFLICT	Use the original payload or a new unique key
422	INVALID_RECIPIENT	Supply a valid E.164 recipient
422	INVALID_SENDER_ID	Supply an active Sender ID assigned to the application
422	MESSAGE_TOO_LONG	Reduce the message to at most 4,096 characters
422	WEBHOOK_CONFIGURATION_INVALID	Correct the URL, secret, or enabled value
429	RATE_LIMIT_EXCEEDED	Back off and retry according to service policy
503	SERVICE_UNAVAILABLE	Retry safely; reuse the same idempotency key
500	INTERNAL_ERROR	Record identifiers and escalate if persistent

INVALID_REQUEST (400) and FORBIDDEN (403) can also occur when the request is malformed or the authenticated principal lacks permission. The complete official error catalog and endpoint mappings are in the [API Reference](#).

10 Verify Your Integration

Complete this check before treating the quick start as successful:

- the message submission returned HTTP 202;
- the response included X-Request-ID and X-Correlation-ID;
- your application persisted `message_id` and `client_reference`;
- replaying the identical submission with the same Idempotency-Key returned

the original message result without creating a second message;

- the status operation returned the same `message_id`;
- the webhook configuration read returns the configured URL with no secret;
- the receiver verifies the timestamp and signature using the raw body;
- the receiver applies a terminal event once and acknowledges it with 2xx;
- logs contain public identifiers but no credential, webhook secret, or

message body.

Status reads remain useful for reconciliation when webhook processing is delayed or unavailable.

11 Next Steps

Use the companion documents according to the task:

Need	Document
Integration architecture, idempotency, security, and production readiness	Developer Integration Guide
Every endpoint, schema, header, status, and error mapping	API Reference
Secure webhook verification and operations	Webhook Integration Guide
Runnable client examples	Repository examples

Before production, complete the production-readiness controls in the Developer Integration Guide, exercise error and replay paths, and retain the request identifiers needed for support and incident investigation.

End of document