



DEVELOPER DOCUMENTATION

SMS API Integration Guide

A complete English reference for your team

Node.js / PHP / Python / C# .NET / Flutter

Send SMS | Read balance | Handle responses | Deploy safely

HTTPS GET / UTF-8 / Plain-text response

Gateway endpoint

<https://sms.masrbokra.com/sendsms.php>

Edition 1.1 | 13 October 2026

Prepared by Together. Based on the supplied production API contract.

01 / INTEGRATION GUIDE

Start here

A practical map for your development team

Together SMS API connects your backend to an SMS gateway using a simple HTTPS GET request. This guide covers sending, balance retrieval, strict response handling and integration examples for Node.js, PHP, Python, .NET and Flutter.

Section	Pages
Setup and architecture	3
API reference: send, balance and responses	4-6
Recipients, encoding and segment planning	7
cURL and Postman	8-9
Node.js backend	10-11
PHP backend	12-13
Python backend	14-15
C# / .NET backend	16-17
Flutter integration through your backend	18-19
Reliability, bulk messaging and OTP	20
Troubleshooting and launch checklist	21
Scope, support and technical references	22

Your first integration

1. Obtain the API username, password and activated Sender ID.
2. Store them on your backend and retrieve the balance first.
3. Send one controlled message to a phone you own.
4. Record gateway acceptance separately from delivery.
5. Add duplicate prevention, safe logging and unknown-outcome handling.

All examples use placeholders or illustrative phone numbers. Do not run a send example until you replace the number with an authorized test recipient. No live SMS requests were made while preparing this document.

02 / INTEGRATION GUIDE

Architecture & configuration

Keep the provider credentials on your server

Layer	Responsibility
Flutter / web application	Authenticate to your backend; request an allowed business action.
Your trusted backend	Authorize, validate, de-duplicate, store operation state and call the gateway.
SMS gateway	Accept or reject the request and return a plain-text body.

Provisioning checklist

Request an active SMS account, API credentials, sufficient balance and an approved, activated Sender ID. Your backend needs outbound HTTPS access to sms.masrbokra.com on port 443 and a valid trusted CA certificate store.

Server-side environment variables

```
MASRBOKRA_SMS_USER=YOUR_USERNAME
MASRBOKRA_SMS_PASSWORD=YOUR_PASSWORD
MASRBOKRA_SMS_SENDER=YOUR_APPROVED_SENDER
```

The MASRBOKRA variable names are used consistently in this guide because the attached contract uses the MasrBokra gateway. Together branding does not change the hostname, credentials, query keys or response formats.

Connection policy

Setting	Recommended policy
Method / encoding	GET / UTF-8, with every query value URL-encoded
Response	Plain text; do not parse as JSON
Timeouts	10-second connect; target 20-second overall deadline
Redirects	Disable automatic redirects
Send retries	No automatic application or queue retries

Never put the provider password into Flutter assets, a mobile build, a browser bundle, a public URL or source control. Disable query-string capture in logs, tracing and monitoring.

03 / INTEGRATION GUIDE

Send SMS

One message body, one or more recipients

```
GET https://sms.masrbokra.com/sendsms.php
```

Parameter	Required value / meaning
user	API username. String; required.
password	API password. String; required.
numbers	International digits; one number or a comma-separated list.
sender	The approved and activated Sender ID assigned to your account.
message	Non-empty UTF-8 message text.
lang	en for English; ar for Arabic / Unicode.

Minimal request, shown with placeholders

```
GET /sendsms.php?user=YOUR_USERNAME&password=YOUR_PASSWORD
&numbers=201001234567&sender=YOUR_APPROVED_SENDER
&message=Your%20order%20is%20ready&lang=en
Host: sms.masrbokra.com
```

Line breaks above are for readability only. The actual URL is one continuous string. Use your HTTP library to encode the individual values; never concatenate unescaped user input.

Multiple recipients

```
numbers=201001234567,201112345678,201512345678
```

Normalize and remove duplicate recipients before building the list. A multi-recipient request shares the same message, sender and language. Personalized messages require separate requests or application-managed grouping of identical messages.

There is no documented POST alternative, JSON body, bearer-token authentication or scheduling field in this contract. Do not add invented parameters. Recipient and URL size limits must be confirmed with Together before large batches.

04 / INTEGRATION GUIDE

Retrieve account balance

Use the same endpoint with action=get

```
GET https://sms.masrbokra.com/sendsms.php
```

Parameter	Required value
user	API username
password	API password
action	get

Request and valid response

```
GET /sendsms.php?user=YOUR_USERNAME&password=YOUR_PASSWORD&action=get
Host: sms.masrbokra.com
```

Example response body:
3956.00

The returned balance is a non-negative numeric plain-text value and may include decimals. Preserve the original value as a string or use a decimal type. Do not truncate to an integer or interpret the value as a currency amount without confirming the account unit.

Conservative validation

```
^[0-9]+(?:\.[0-9]+)?
```

First require an HTTP 2xx response, then trim surrounding whitespace and validate the entire body. This validator accepts 0, 3956.00 and 120.75. Reject negative values, NaN, Infinity, HTML, empty bodies and error messages.

Operational use

Read the balance before onboarding tests and monitor it during production. A balance read does not reserve credit: another send may consume balance between your check and your request. The gateway balance is authoritative; a local segment calculation is an estimate.

Balance retrieval is read-only and can use a small bounded retry policy. A balance difference cannot prove whether a particular timed-out send succeeded, especially when other sends are running.

05 / INTEGRATION GUIDE

Responses & error handling

Require both HTTP success and a valid response body

Exact trimmed body	Interpretation / action
1	Gateway accepted the request.
1:201001234567	Accepted; gateway returned a recipient.
1:201001234567,201112345678	Accepted aggregate acknowledgement.
0	General rejection / failure.
Invalid login	Check credentials securely.
Missing or Empty Field	Check every required value.
Insufficient Balance	Recharge or reduce the send.
Invalid Sender	Verify the Sender ID value.
Sender name is not activated	Complete Sender ID activation.
Invalid Message Language	Correct lang and message encoding.
Anything else / empty body	Unknown outcome; investigate before resending.

Strict acknowledgement pattern

```
1 OR full match: 1:[1-9][0-9]{7,14}(?:,[1-9][0-9]{7,14})*
```

Use full-string matching after trim. Never use `startsWith("1")`, a `truthy-body` check or `HTTP 200` alone. Responses such as `10`, `11:201...`, `1:`, `1:abc` or `HTML` must not count as accepted.

Separate the three outcomes

Accepted: a strict acknowledgement was received.

Rejected: a known rejection body was received with `HTTP 2xx`.

Unknown: timeout, transport error, `HTTP non-2xx`, empty or undocumented body. The request may already have reached the gateway.

Acceptance is not confirmed handset delivery. No message ID or delivery-report endpoint is documented. If returned recipients differ from the requested list, flag the discrepancy for review; do not infer per-number delivery or blindly resend missing numbers.

06 / INTEGRATION GUIDE

Recipients, encoding & segments

Validate the number and plan the message cost

Local Egyptian input	Value sent to the gateway
01001234567	201001234567
01112345678	201112345678
01212345678	201212345678
01512345678	201512345678

For the Egyptian reference integration, accept 11-digit local numbers starting 010, 011, 012 or 015; replace the leading 0 with 20. For an already international number, use 8-15 ASCII digits starting 1-9, without +, spaces or punctuation. Format validity does not guarantee a reachable or permitted destination; confirm supported countries for your account.

Text and URL encoding

Use lang=en for English and lang=ar for Arabic or Unicode content. Pass the original UTF-8 text to the query builder once. Encoding each value keeps ampersands, plus signs, percent signs, spaces and line breaks from changing the request structure. Do not pre-encode and then encode again.

Planning profile	Single SMS	Multipart size
English / GSM-7 baseline	160	153 per segment
Arabic / Unicode baseline	70	67 per segment

```
segments = 1 if length <= single_limit
           else ceil(length / multipart_limit)
estimated units = segments * unique_recipient_count
```

Examples: 161 basic English characters estimate 2 segments; 71 Unicode units estimate 2 segments. Two segments to three unique numbers estimate 6 SMS units.

These are planning baselines, not a gateway billing guarantee. GSM extension characters can consume extra units; emoji and Unicode pairs can affect counting. The reference integration counts a newline as two characters. Test real content and confirm billing rules; lang=en is not proof of one unit per character.

07 / INTEGRATION GUIDE

Quick test with cURL

Server-side terminal examples

Set the three environment variables on page 3 before testing. Replace the illustrative phone number with your own authorized test device. Run a balance request first.

Read the balance

```
curl --get 'https://sms.masrbokra.com/sendsms.php' \  
  --connect-timeout 10 --max-time 20 \  
  --data-urlencode "user=${MASRBOKRA_SMS_USER}" \  
  --data-urlencode "password=${MASRBOKRA_SMS_PASSWORD}" \  
  --data-urlencode 'action=get' \  
  --write-out '\nHTTP_STATUS=%{http_code}\n'
```

Send one English message

```
curl --get 'https://sms.masrbokra.com/sendsms.php' \  
  --connect-timeout 10 --max-time 20 \  
  --data-urlencode "user=${MASRBOKRA_SMS_USER}" \  
  --data-urlencode "password=${MASRBOKRA_SMS_PASSWORD}" \  
  --data-urlencode 'numbers=201001234567' \  
  --data-urlencode "sender=${MASRBOKRA_SMS_SENDER}" \  
  --data-urlencode 'message=Your order is ready' \  
  --data-urlencode 'lang=en' \  
  --write-out '\nHTTP_STATUS=%{http_code}\n'
```

The HTTP_STATUS line is produced locally by cURL; it is not part of the gateway response. Evaluate the gateway body separately using page 6. A zero cURL exit code alone does not establish acceptance.

For Unicode messages

Read your UTF-8 text from a private file with `--data-urlencode 'message@message.txt'` and set `lang=ar`. Treat the file as sensitive and remove it after the controlled test.

Do not use verbose tracing, `--location` or `--retry` on send requests. Shell argument inspection can expose expanded credentials; these commands are controlled diagnostic examples. Use your backend and secrets manager for production.

08 / INTEGRATION GUIDE

Postman & mock testing

Inspect the body without exposing credentials

Create a GET request to <https://sms.masrbokra.com/sendsms.php>. Set parameters in the Params editor and let Postman encode them. Use private local secrets; export only placeholders.

Send parameter	Value
user / password / sender	{{MASRBOKRA_SMS_USER}} / {{MASRBOKRA_SMS_PASSWORD}} / {{MASRBOKRA_SMS_SENDER}}
numbers	Your authorized test number in international digits
message	Test message
lang	en

For balance, remove the send-only parameters and keep user, password and action=get. Disable automatic redirect following. Set a request timeout and do not configure automated send retries.

Acceptance assertion for a controlled send

```
const body = pm.response.text().trim();
pm.test('Gateway accepted this request', () => {
  pm.expect(pm.response.code >= 200 &&
    pm.response.code < 300).to.eql(true);
  const ok = body === '1' ||
    /^1:[1-9][0-9]{7,14}(?:,[1-9][0-9]{7,14})*$/.test(body);
  pm.expect(ok).to.eql(true);
});
```

Test the parser locally before a live send

Fixture	Expected state
1 / 1:201001234567	accepted
0 / Invalid login / Insufficient Balance	rejected
10 / 1: / 1:abc / empty / HTML	unknown
HTTP 302 / 500 / timeout	unknown

Postman console, history and exports may contain the expanded credential-bearing URL. Do not share them. Inspect mock fixtures locally and use live sends only for deliberate, limited validation.

09 / INTEGRATION GUIDE

Node.js | reusable client

Native fetch; ES module; server-side only

Use a supported Node.js runtime with native fetch and AbortSignal.timeout. This module has a 20-second deadline covering fetch and response reading; it does not separately set a 10-second connect timeout.

```
// together-sms.mjs - trusted backend only
const endpoint = 'https://sms.masrbokra.com/sendsms.php';
const known = new Set([
  '0', 'Invalid login', 'Missing or Empty Field',
  'Insufficient Balance', 'Invalid Sender',
  'Sender name is not activated', 'Invalid Message Language'
]);
function secret(name) {
  const value = process.env[name];
  if (!value) throw new Error('Missing SMS configuration');
  return value;
}
async function call(params) {
  const url = new URL(endpoint);
  url.search = new URLSearchParams({
    user: secret('MASRBOKRA_SMS_USER'),
    password: secret('MASRBOKRA_SMS_PASSWORD'), ...params
  }).toString();
  try {
    const response = await fetch(url, {
      method: 'GET', redirect: 'error',
      signal: AbortSignal.timeout(20_000)
    });
    if (!response.ok) throw new Error();
    return (await response.text()).trim();
  } catch {
    throw new Error('SMS transport failure; outcome unknown');
  }
}
export async function sendSms(numbers, message, lang = 'en') {
  const body = await call({ numbers, message, lang,
    sender: secret('MASRBOKRA_SMS_SENDER') });
  const accepted = body === '1' ||
    /^1:[1-9][0-9]{7,14}(?:,[1-9][0-9]{7,14})*$/.test(body);
  return { state: accepted ? 'accepted' :
    known.has(body) ? 'rejected' : 'unknown' };
}
export async function getBalance() {
  const body = await call({ action: 'get' });
  if (!/^([0-9]+(?:\.[0-9]+)?)/.test(body))
    throw new Error('Invalid balance response');
  return body; // Preserve decimal precision as text.
}
```

No package dependency is required for this module. Configure the three environment variables before calling it. Never log url or raw network exceptions.

10 / INTEGRATION GUIDE

Node.js | send & balance

Call the module from your existing backend

```
// demo.mjs - importing the module does not send a message
import { sendSms, getBalance } from './together-sms.mjs';

function egyptianRecipients(input) {
  const values = input.map(value => {
    const n = String(value).trim();
    if (/^01[0125][0-9]{8}$/.test(n)) return '20' + n.slice(1);
    if (/^201[0125][0-9]{8}$/.test(n)) return n;
    throw new Error('Invalid Egyptian mobile number');
  });
  return [...new Set(values)].join(',');
}

const balance = await getBalance(); // Decimal string.
const numbers = egyptianRecipients(['01001234567', '01001234567']);
// Replace this number with your authorized test recipient first.
try {
  const result = await sendSms(numbers, 'Your order is ready', 'en');
  if (result.state === 'accepted') {
    console.log('Gateway accepted');
  } else if (result.state === 'rejected') {
    console.log('Gateway rejected');
  } else {
    console.log('Outcome unknown; review before resending');
  }
} catch {
  console.log('Outcome unknown; do not automatically retry');
}
```

Framework integration

In Express, NestJS or another backend framework, call `sendSms` only after authentication, authorization, request validation and a durable duplicate check. Keep sender, credentials and the gateway URL in trusted configuration. Validate message length and lang before the call.

For order notifications, derive the recipient from the authorized order record. For OTP, generate the code and message on the server. A public endpoint that accepts arbitrary recipients and messages becomes an SMS relay and can consume your balance.

Do not add a general HTTP retry interceptor or a queue retry policy around `sendSms`. Persist unknown outcomes for review. Return only your own safe application status to Flutter or the browser.

11 / INTEGRATION GUIDE

PHP | reusable client

PHP 8+ with the cURL extension

Save as `together_sms.php`. TLS verification stays enabled; redirects are disabled. cURL applies a 10-second connect timeout and a 20-second overall transfer timeout.

```
<?php // together_sms.php - trusted backend only
declare(strict_types=1);

function smsSecret(string $key): string {
    $value = getenv($key);
    if ($value === false || $value === '') {
        throw new RuntimeException('Missing SMS configuration');
    }
    return $value;
}

function smsCall(array $params): string {
    $params = array_merge([
        'user' => smsSecret('MASRBOKRA_SMS_USER'),
        'password' => smsSecret('MASRBOKRA_SMS_PASSWORD'),
    ], $params);
    $query = http_build_query($params, '', '&', PHP_QUERY_RFC3986);
    $ch = curl_init('https://sms.masrbokra.com/sendsms.php?'.$query);
    if ($ch === false) throw new RuntimeException('Client error');
    curl_setopt_array($ch, [
        CURLOPT_RETURNTRANSFER => true,
        CURLOPT_CONNECTTIMEOUT => 10,
        CURLOPT_TIMEOUT => 20,
        CURLOPT_FOLLOWLOCATION => false,
        CURLOPT_SSL_VERIFYPEER => true,
        CURLOPT_SSL_VERIFYHOST => 2,
    ]);
    $body = curl_exec($ch);
    $status = (int) curl_getinfo($ch, CURLINFO_RESPONSE_CODE);
    curl_close($ch);
    if ($body === false || $status < 200 || $status >= 300) {
        throw new RuntimeException('Transport failure; outcome unknown');
    }
    return trim($body);
}

function smsState(string $body): string {
    if ($body === '1' || preg_match(
        '/\A1:[1-9][0-9]{7,14}(?:,[1-9][0-9]{7,14})*\z/', $body
    ) === 1) return 'accepted';
    $known = ['0', 'Invalid login', 'Missing or Empty Field',
        'Insufficient Balance', 'Invalid Sender',
        'Sender name is not activated', 'Invalid Message Language'];
    return in_array($body, $known, true) ? 'rejected' : 'unknown';
}
```

Do not log `curl_error()`, the request URL, query parameters or raw request traces. Keep deployment error pages and logs free of provider secrets.

12 / INTEGRATION GUIDE

PHP | send & balance

Use the same helper for both operations

```
<?php // demo.php - run only after configuring a test recipient
require __DIR__.'/together_sms.php';

function smsSend(string $numbers, string $message, string $lang): string {
    return smsState(smsCall([
        'numbers' => $numbers, 'message' => $message, 'lang' => $lang,
        'sender' => smsSecret('MASRBOKRA_SMS_SENDER'),
    ]));
}

function smsBalance(): string {
    $body = smsCall(['action' => 'get']);
    if (preg_match('/\A[0-9]+(?:\.[0-9]+)?\z/', $body) !== 1) {
        throw new RuntimeException('Invalid balance response');
    }
    return $body; // Do not cast to int or float for accounting.
}

try {
    $balance = smsBalance();
    // Normalize and de-duplicate before joining this array.
    $numbers = implode(',', array_unique(['201001234567']));
    $state = smsSend($numbers, 'Your order is ready', 'en');
    echo match ($state) {
        'accepted' => 'Gateway accepted',
        'rejected' => 'Gateway rejected',
        default => 'Outcome unknown; review before resending',
    };
} catch (RuntimeException $e) {
    // Do not output the exception, stack trace or request URL.
    echo 'Operation could not be confirmed; no automatic retry';
}
```

Laravel and other PHP frameworks

Inject this adapter into a server-side service and call it from an authorized controller or worker after validation. Keep secrets in configuration; prevent duplicate jobs and automatic retries after uncertainty.

Deployment checks

Ensure ext-curl is installed, a valid CA bundle is available and the source file is UTF-8. If TLS fails, update the CA bundle rather than disabling certificate checks.

For decimal arithmetic, use a decimal-capable library or BCMath. This example returns the original balance string and does not assume the provider balance is a money amount.

13 / INTEGRATION GUIDE

Python | reusable client

requests-based adapter; server-side only

Install the dependency with `python -m pip install requests` and pin it in your project. `requests timeout=(10, 20)` means connect and read-inactivity timeouts, not a 20-second wall-clock deadline.

```
# together_sms.py - trusted backend only
import os
import re
import requests
ENDPOINT = 'https://sms.masrbokra.com/sendsms.php'
ACK = re.compile(r'1:[1-9][0-9]{7,14}(?:,[1-9][0-9]{7,14})*')
KNOWN = {
    '0', 'Invalid login', 'Missing or Empty Field',
    'Insufficient Balance', 'Invalid Sender',
    'Sender name is not activated', 'Invalid Message Language',
}

def secret(key):
    value = os.environ.get(key)
    if not value:
        raise RuntimeError('Missing SMS configuration')
    return value

def call(params):
    query = {
        'user': secret('MASRBOKRA_SMS_USER'),
        'password': secret('MASRBOKRA_SMS_PASSWORD'), **params,
    }
    try:
        response = requests.get(
            ENDPOINT, params=query, timeout=(10, 20),
            allow_redirects=False,
        )
        if not 200 <= response.status_code < 300:
            raise requests.RequestException()
        response.encoding = 'utf-8'
        return response.text.strip()
    except requests.RequestException:
        raise RuntimeError(
            'SMS transport failure; outcome unknown'
        ) from None

def send_sms(numbers, message, lang='en'):
    body = call({
        'numbers': numbers, 'message': message, 'lang': lang,
        'sender': secret('MASRBOKRA_SMS_SENDER'),
    })
    if body == '1' or ACK.fullmatch(body):
        return 'accepted'
    return 'rejected' if body in KNOWN else 'unknown'

def get_balance():
    body = call({'action': 'get'})
    if not re.fullmatch(r'[0-9]+(?:\.[0-9]+)?', body):
        raise RuntimeError('Invalid balance response')
    return body # Keep precision; convert with Decimal if needed.
```

Enforce hard deadlines in your worker; cancellation does not prove rejection.

14 / INTEGRATION GUIDE

Python | send & balance

Decimal precision and controlled outcome handling

```
# demo.py - configure a test recipient before running
from decimal import Decimal
import re
from together_sms import get_balance, send_sms

def egyptian_recipients(values):
    result = []
    for value in values:
        n = str(value).strip()
        if re.fullmatch(r'01[0125][0-9]{8}', n):
            n = '20' + n[1:]
        elif not re.fullmatch(r'201[0125][0-9]{8}', n):
            raise ValueError('Invalid Egyptian mobile number')
        if n not in result:
            result.append(n)
    if not result:
        raise ValueError('No recipients')
    return ','.join(result)

balance_text = get_balance()
balance = Decimal(balance_text) # Exact decimal representation.
numbers = egyptian_recipients(['01001234567', '01001234567'])
try:
    state = send_sms(numbers, 'Your order is ready', 'en')
    if state == 'accepted':
        print('Gateway accepted')
    elif state == 'rejected':
        print('Gateway rejected')
    else:
        print('Outcome unknown; review before resending')
except RuntimeError:
    print('Outcome unknown; do not automatically retry')
```

Django, Flask or FastAPI

Keep the adapter in a service module and call it from a protected business flow. The requests client is synchronous: run it in an appropriate worker or thread when your framework uses an asynchronous event loop. Keep SMS processing bounded and persist operation state before contacting the gateway.

Worker and exception policy

Disable send retries and task replay. Exclude HTTP request details from monitoring; the sample suppresses raw exception chaining.

A balance request can be retried with a separate bounded policy. Do not apply that policy to send_sms or to a transaction that already attempted a send.

15 / INTEGRATION GUIDE

C# / .NET | client setup

C# 10+ on a supported .NET runtime

Create TogetherSms.cs with the code below and append page 17. The static HttpClient reuses connections, disables redirects and sets connect and overall timeouts.

```
// TogetherSms.cs - same file continues on the next page
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Http;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
public static class TogetherSms
{
    private const string Endpoint =
        "https://sms.masrbokra.com/sendsms.php";
    private static readonly HttpClient Client = new(
        new SocketsHttpHandler {
            AllowAutoRedirect = false,
            ConnectTimeout = TimeSpan.FromSeconds(10)
        }) { Timeout = TimeSpan.FromSeconds(20) };
    private static string Secret(string key) =>
        Environment.GetEnvironmentVariable(key) is { Length: > 0 } v
        ? v : throw new InvalidOperationException("Missing SMS config");
    private static async Task<string> Call(
        Dictionary<string, string> parameters)
    {
        parameters["user"] = Secret("MASRBOKRA_SMS_USER");
        parameters["password"] = Secret("MASRBOKRA_SMS_PASSWORD");
        var query = string.Join("&", parameters.Select(p =>
            Uri.EscapeDataString(p.Key) + "=" +
            Uri.EscapeDataString(p.Value)));
        try {
            using var response = await Client.GetAsync(Endpoint+"?" + query);
            if (!response.IsSuccessStatusCode)
                throw new HttpRequestException();
            return (await response.Content.ReadAsStringAsync()).Trim();
        }
        catch (Exception e) when (e is HttpRequestException
            or TaskCanceledException) {
            throw new InvalidOperationException(
                "SMS transport failure; outcome unknown");
        }
    }
}
```

In ASP.NET Core, an injected HttpClient service is also suitable. Redact HTTP client logging and application telemetry before enabling production traffic; request URIs carry the credentials and message.

No external package is required for this sample. Do not attach a resilience handler that retries sends automatically.

16 / INTEGRATION GUIDE

C# / .NET | send & balance

Finish the client and invoke it from your server

```
public static async Task<string> SendSms(
    string numbers, string message, string lang = "en")
{
    var body = await Call(new() {
        ["numbers"] = numbers, ["message"] = message,
        ["lang"] = lang,
        ["sender"] = Secret("MASRBOKRA_SMS_SENDER")
    });
    if (body == "1" || Regex.IsMatch(body,
        @"\A1:[1-9][0-9]{7,14}(?:,[1-9][0-9]{7,14})*\z"))
        return "accepted";
    var known = new[] { "0", "Invalid login",
        "Missing or Empty Field", "Insufficient Balance",
        "Invalid Sender", "Sender name is not activated",
        "Invalid Message Language" };
    return known.Contains(body) ? "rejected" : "unknown";
}
public static async Task<string> GetBalance()
{
    var body = await Call(new() { ["action"] = "get" });
    if (!Regex.IsMatch(body, @"\A[0-9]+(?:\.[0-9]+)?\z"))
        throw new InvalidOperationException("Invalid balance response");
    return body;
}
} // End of TogetherSms.cs
```

Program.cs or an authorized service method

```
// Replace the illustrative number before any live send.
var balanceText = await TogetherSms.GetBalance();
try {
    var state = await TogetherSms.SendSms(
        "201001234567", "Your order is ready", "en");
    Console.WriteLine(state); // Safe enum-like state only.
} catch (InvalidOperationException) {
    Console.WriteLine("Could not confirm; no automatic retry");
}
```

Validate and de-duplicate recipients before calling SendSms. Keep the balance as text; if converting to decimal, use invariant culture and handle values outside the selected numeric type. Store accepted, rejected and unknown as separate application states.

In ASP.NET, authenticate and authorize the business action before sending. Never return the gateway URI or a raw HttpRequestException to the caller.

17 / INTEGRATION GUIDE

Flutter | integration design

Flutter calls your backend; your backend calls the gateway

A mobile application cannot safely hold the SMS gateway password. The Flutter app should request a business action from your own authenticated HTTPS API. Use any of the server adapters on pages 10-17 behind that API.

Suggested private application endpoint

```
POST https://YOUR_BACKEND_HOST/api/orders/{orderId}/notify
Authorization: Bearer <YOUR_APPLICATION_ACCESS_TOKEN>
Content-Type: application/json

{
  "requestId": "<unique ID for this business operation>"
}
```

This POST endpoint and its JSON responses are an application design example that you must implement. They are not provided by the SMS gateway. No provider username, password, sender or message is submitted by Flutter.

Backend step	Required behavior
Authenticate / authorize	Confirm the caller can notify this order.
Derive the message	Load recipient and permitted text from server records.
Check duplicates	Persist a unique requestId scoped to caller and operation.
Call gateway once	Send the normalized number, approved sender and lang.
Store result	accepted, rejected or unknown; never label it delivered.
Return safe JSON	Expose only your operationId and application status.

Suggested backend response contract

```
HTTP 200: {"operationId":"op_example","status":"accepted"}
HTTP 200: {"operationId":"op_example","status":"rejected"}
HTTP 200: {"operationId":"op_example","status":"unknown"}
HTTP 401/403: caller not authenticated / not permitted
HTTP 400: invalid order or request identifier
```

Make requestId persistent and unique in your database. Repeated requests with the same key should return the existing operation; they must not call the gateway again. A local duplicate key is not a gateway idempotency key.

18 / INTEGRATION GUIDE

Flutter | Dart example

Consume the private backend contract on page 18

```
// Add dependency: flutter pub add http
import 'dart:async';
import 'dart:convert';
import 'package:http/http.dart' as http;

Future<String> notifyOrder({
  required String orderId,
  required String requestId,
  required String appAccessToken,
}) async {
  final uri = Uri(scheme: 'https', host: 'YOUR_BACKEND_HOST',
    pathSegments: ['api', 'orders', orderId, 'notify']);
  final client = http.Client();
  try {
    final request = http.Request('POST', uri)
      ..followRedirects = false
      ..headers.addAll({
        'Authorization': 'Bearer $appAccessToken',
        'Content-Type': 'application/json',
      })
      ..body = jsonEncode({'requestId': requestId});
    final response = await (() async {
      final stream = await client.send(request);
      return http.Response.fromStream(stream);
    })().timeout(const Duration(seconds: 30));
    if (response.statusCode != 200) {
      throw StateError('Notification could not be confirmed');
    }
    final data = jsonDecode(utf8.decode(response.bodyBytes));
    if (data is! Map<String, dynamic> ||
      !{'accepted', 'rejected', 'unknown'}.contains(data['status'])) {
      throw StateError('Unexpected application response');
    }
    return data['status'] as String;
  } catch (_) {
    // Neither a timeout nor an app error proves that no SMS was sent.
    throw StateError('Check operation status before another attempt');
  } finally {
    client.close();
  }
}
```

Create and persist requestId when the user starts the operation; reuse it when checking or recovering that operation. Disable the submit button while pending. Keep the application access token in your normal authenticated-session flow and never log it.

On a client timeout, check the saved operation through your own backend status route. Implement that route as read-only. Future.timeout stops waiting; it does not prove that your backend stopped processing.

19 / INTEGRATION GUIDE

Reliable production flows

Prevent duplicate sends and handle uncertainty explicitly

Operation lifecycle

State	Meaning / next step
pending	Persist the business operation and acquire a unique send lock.
sending	A worker is making the request; prevent concurrent dispatch.
accepted	Gateway acknowledgement received; release lock and record acceptance.
rejected	Known rejection received; correct the cause before a new deliberate send.
unknown	May be accepted; hold for review. Do not auto-resend.

Persist sending before network dispatch. If a worker crashes in this state, recover it as unknown for review rather than automatically replaying the job. A timeout or proxy HTTP 500 can occur after the gateway has processed the request.

Bulk and personalized messaging

Normalize and de-duplicate globally before splitting recipients into batches. Obtain the maximum URL length, recipients per request, supported countries and rate policy from Together; no numerical limit is documented. Batch under those confirmed limits and use bounded concurrency. Persist every batch state separately.

This GET contract sends one text to all listed recipients. Resolve template variables in your backend and group only recipients with identical final content. Do not send unresolved {{name}} placeholders expecting the gateway to expand them.

OTP and scheduled notifications

Generate OTPs securely on the backend, with expiry, limited verification attempts and resend cooldowns. Store a verification representation instead of logging the raw code. The SMS API does not generate or verify OTPs. Build scheduling in your own backend queue; no scheduling parameter is documented.

Safe telemetry and balance accounting

Log operation ID, timestamp, state, duration and recipient count. Exclude query strings, message text, secrets and raw exceptions. Retain uncertain credit reservations for review; reconcile with the provider before refunding or deducting again.

Do not invent delivery tracking. Without a documented message ID or DLR, your application can report gateway acceptance and observed test delivery separately.

20 / INTEGRATION GUIDE

Troubleshooting & go-live

A checklist your developers can use before release

Symptom	Check / next step
Invalid login	Verify account credentials without printing them.
Missing or Empty Field	Check startup configuration and all send fields.
Insufficient Balance	Read live balance; confirm required segment units.
Invalid Sender / not activated	Confirm exact approved Sender ID and activation.
Invalid Message Language	Use ar for Unicode; confirm UTF-8 and single encoding.
Corrupted or truncated text	Check query encoding, source-file encoding and URL limits.
Timeout / HTTP non-2xx	Mark unknown; do not blindly resend.
HTTP 2xx but strange body	Require a complete acknowledgement match; review safely.
Duplicate notifications	Check button, job, operation and recipient de-duplication.
Balance differs from estimate	Review Unicode, multipart length and concurrent sends.

Before your first production release

1. Confirm credentials, active sender, balance and outbound HTTPS.
2. Confirm recipient, URL and rate limits with Together.
3. Verify English and Unicode content using authorized devices.
4. Test accepted, rejected, unknown and HTTP failure fixtures.
5. Verify balance decimals and empty / invalid balance handling.
6. Confirm no secrets appear in application, proxy or tracing logs.
7. Disable redirect following and all automatic send retries.
8. Verify duplicate keys across concurrent and repeated requests.
9. Test worker crash recovery and unknown-outcome review.
10. Present acceptance accurately; obtain one controlled delivery observation.

The code examples are integration starting points. Add your project-specific validation, authorization, persistence and monitoring. They were reviewed against the supplied contract; no account-level or live gateway validation is represented by this document.

21 / INTEGRATION GUIDE

Scope, support & references

Together technical documentation | Edition 1.1

Contract scope

This English edition is based on the supplied MasrBokra SMS HTTP API Integration Guide v1.0, dated 27 August 2026. Prepared for Together on 13 October 2026. The examples and application architecture have been expanded; the documented provider endpoint and wire contract remain unchanged.

The supplied contract does not define a message ID, DLR endpoint, per-recipient delivery result, idempotency key, scheduling field, guaranteed safe retry, sandbox endpoint or fixed throughput/batch limit. Obtain written confirmation before designing around additional capabilities.

Account and integration support

Contact your assigned Together account or support representative for credentials, Sender ID approval, balance, account coverage, confirmed limits and provider-side questions. Share the operation timestamp with timezone, safe operation ID, HTTP status and a sanitized response description. Do not share a full request URL, password, message text or unmasked recipient list.

Technical references

Provider contract: supplied integration guide v1.0 (27 August 2026). This is the source for the endpoint, parameters, acknowledgement formats and known error strings.

Runtime references below support the client implementation notes; they do not define Together gateway behavior.

Reference	Official documentation
Node.js fetch / AbortSignal	https://nodejs.org/api/globals.html
PHP query encoding	https://www.php.net/manual/en/function.http-build-query.php
PHP cURL configuration	https://www.php.net/manual/en/function.curl-setopt.php
Python Requests timeouts	https://requests.readthedocs.io/en/stable/user/advanced/
.NET HTTP handler	https://learn.microsoft.com/dotnet/api/system.net.http.socketshttphandler
Flutter HTTP networking	https://docs.flutter.dev/cookbook/networking/send-data
SMS encoding background	https://www.twilio.com/docs/glossary/what-sms-character-limit

Use supported runtimes and pinned dependencies in your application. Technical reference links were checked when this edition was prepared. All documentation text and example comments in this edition are English.