2Factor API Documentation

2Factor provides a robust and scalable HTTP API platform for sending SMS OTPs, transactional messages, and promotional bulk SMS efficiently.

In addition to SMS services, we also provide advanced WhatsApp solutions: WhatsApp Chatbot Provider, WhatsApp Business API Guide 2025, and WhatsApp API Service Provider in India.

Authentication

2Factor employs API key-based authentication to ensure secure access. Each client is assigned a unique `APIKey` that identifies their account.

Additional Security Features:

- **IP-Based Restrictions**: To enhance security, clients can restrict API access to a specific set of IP addresses.

API Throughput and Scalability

The **2Factor Cloud API** is designed for enterprise-grade performance and reliability, trusted by leading organizations.

Default Performance

- **Throughput**: Supports up to **200 requests per second** under default configurations.
- **Response Times**: Average response times range from **100 ms to 500 ms**.

High Availability (HA) Cluster for Enhanced Scalability

For clients with high scalability needs, we offer a **High Availability (HA) Cluster** infrastructure:

- **Throughput**: Supports up to **2000 transactions per second (TPS)**.
- **Response Times**: Average response times range from **50ms to 150 ms**.

Scalability Requests

To scale up or discuss your specific requirements, contact **support@2factor.in**.

Data Privacy & Security Measures

At **2Factor**, safeguarding customer data is a top priority. The following measures ensure that all data is securely processed, stored, and accessed in compliance with industry standards:

Security Measures

API Hosting and Encryption

- **Cloudflare Serverless Environment**: All **2Factor API endpoints** are hosted on Cloudflare's serverless environment for scalability.
- **End-to-End Encryption**: Cloudflare employs **SSL/TLS encryption** to ensure secure transmission of data between the client, Cloudflare, and 2Factor's backend systems.

Threat Intelligence and Protection

- **Web Application Firewall (WAF)**: Cloudflare's WAF protects publicly accessible API endpoints from vulnerabilities and unauthorized access attempts.
  - **Rule Set**: Protects against thousands of commonly known web exploits and intrusion attempts.
  - **IP-Based Restrictions**: Backend systems enforce strict IP-based access controls to block unintended audiences.

Brute Force and Local Firewall Protection

- Each Linux virtual machine (VM) hosting code and data stores is secured with:
  - A **local firewall**.

    - **Brute-force monitoring software** to detect and prevent unauthorized login attempts.

Logging and Monitoring

- Random checks are conducted to monitor access attempts to data stores or VMs from unauthorized IPs, ensuring continuous oversight of system security.

Additional Security Features for Clients

IP-Based Access Control

- **Client-Specific IP Restrictions**: Clients can enforce IP-based restrictions on their API accounts to prevent unauthorized use.

Data Retention and Disposal Policies

Data Retention

- **Banking Clients**: Data is retained for **7 years**, as required by industry regulations.
- **Other Clients**: Data is typically retained for **1 year**, after which it is archived in flat files.

Data Archival and Disposal

- After the retention period, data is dumped into **SQL flat files** and stored in **S3/S3-compatible cold storage**.
- While no formal archival/disposal agreements exist, the above process ensures long-term data security and compliance.

Access to SMS Logs

Multi-Factor Authentication

- To access SMS logs, users must complete **2-factor authentication**:
    - **Step 1**: Log in with a username and password.

    - **Step 2**: Verify the login attempt using an SMS OTP sent to the registered phone number.

Access Control

- **Account Owners**: Full access to SMS logs.
- **2Factor Technical Support Team**: Limited access on a **need-to-know basis** to resolve support tickets.
- **Third Parties**: No access is provided to external entities or unauthorized users.

Contact

For more information about 2Factor’s security measures, data retention policies, or access controls, reach out to our support team at support@2factor.in.

Getting Started

1: Create a 2Factor Account

How to Create a Free Trial Account on 2Factor

To get started with **2Factor**, follow these steps:

Requirements for Free Trial Signup

1. **Email ID**: Your valid email address.
2. **Phone Number**: A phone number to register your account.

Steps to Create a Free Trial Account

1. Visit the **2Factor signup page**.
2. Enter your **Email ID** and **Phone Number** to register.
3. Upon successful registration:
   a. You will receive **250 free credits** to test and integrate our services.
   b. Account details will be sent to your **registered email address**.

Preparing to Send SMS via 2Factor

Before you can send SMS messages, ensure the following prerequisites are ready:

1. **Sender ID/Header**:
   a. A unique, **6-character identifier** representing your company or brand name.
   b. Example: `MYBRND`.
2. **SMS Template**:
   a. The predefined text message you intend to send.
   b. Templates must comply with DLT guidelines.

Requesting Sender ID and Template Approval

1. Log in to the **2Factor Dashboard**.
2. Navigate to:
   a. **Transactional SMS > Manage Sender IDs > Request Sender ID/Template**.
3. Submit your Sender ID and SMS Template for approval.

- **Approval Time**: Templates are usually approved within **30 minutes to 1 hour** during working hours.
- Once approved:
  - You will receive an **email notification**.
  - The approved Sender ID and Template, along with the relevant **API endpoint**, will be shared.

Begin API Testing

Once your Sender ID and Template are approved:

- You can start testing the **2Factor API** immediately.
- API documentation and credentials will be provided in your email.

DLT Registration for SMS Delivery

To successfully deliver SMS messages, you must complete the **DLT Registration process**. This process ensures compliance with regulatory requirements. For more details on DLT registration, refer to **Step 2** of the documentation.

If you have any questions or need assistance, contact us at support@2factor.in.

2: Complete DLT Registration

Overview of DLT Registration

The Telecom Regulatory Authority of India (TRAI) mandates that all entities engaging in commercial communications register on the Distributed Ledger Technology (DLT) platform. This initiative aims to curb Unsolicited Commercial Communication (UCC) and enhance mobile subscriber privacy in India. Detailed information about these regulations is available in the Telecom Commercial Communications Customer Preference Regulation, 2018 (TCCCPR 2018)

Purpose of These Changes

The primary objective is to reduce the incidence of fraud and spam via SMS and voice calls. By conducting Know Your Customer (KYC) processes with telecom operators, end-to-end traceability is ensured in cases of fraudulent activities. Additionally, pre-defining SMS content helps control spam and prevent deceptive messages.

Enterprises vs. Telemarketers

The DLT platform categorizes registrants as either Enterprises or Telemarketers:

- **Enterprises**: Commercial businesses (individuals or companies) offering products or services and seeking to communicate with customers via SMS or voice services.

- **Telemarketers**: Entities that facilitate communication on behalf of enterprises. They are further classified into:
    - **Telemarketer Aggregators**: Individuals or companies that collect SMS/voice traffic from enterprises without a direct connection to telecom operators.
    - **Telemarketer Delivery**: Individuals or companies that collect SMS/voice traffic from enterprises and have a direct connection with telecom operators.

Note: You or your clients must register on the DLT platform as an **Enterprise**. Solv Technologies (2Factor) will act as the Telemarketer on your behalf. Our Telemarketer Registration Number is (1002884117264848 - SOLV TECHNOLOGIES)

How to Register on the DLT Platform

If you intend to send SMS messages to users in India, DLT registration is mandatory. Registration with at least one telecom operator is required.

We recommend starting with Vodafone's DLT platform, as their support is prompt and approval times are minimal. Alternatively, you may choose other telecom operators' DLT portals.

- **Vodafone DLT Registartion Guide**
- Airtel DLT Registration Guide
- Videocon DLT Registration Guide

For further assistance, feel free to contact us at support@2factor.in.

3: Link Your DLT Account with 2Factor

After your SMS templates have been approved on the **DLT Portal**, the next step is to map them to your **2Factor account**. This ensures seamless integration between your DLT-registered templates and 2Factor's Cloud SMS API.

Steps to Link Your DLT Account

1. **Log in to the 2Factor DLT Portal (** https://dlt-registration.2factor.in)
    a. Navigate to the **DLT Registration** section under **"My DLT Registrations"**.

2. **Update Your DLT Details**:
    a. Enter the **DLT Principal Entity ID (PE ID)** associated with your approved templates.
    b. Ensure that all required fields, such as your DLT registration number and sender ID, are correctly updated.
3. **Mapping Process**:
    a. Once your details are submitted, 2Factor will map your templates to its system.
    b. This process typically takes a few hours to complete.
4. **Account Activation**:
    a. After the mapping is finalized, your **Cloud SMS API account** will be activated.
    b. You will receive an email notification confirming the completion of the setup.
5. **Send a Test Message**:
    a. Use your mapped templates to send a test SMS via the 2Factor API.

Important Notes

- Ensure the **DLT templates** and **sender IDs** you submit match those approved by your telecom operator.
- Mapping delays are minimal but may vary depending on template and sender ID validation.

Support

For assistance with DLT account mapping or any other inquiries, contact us at support@2factor.in.

Send SMS OTP

SMS OTP API Endpoints: Sending and Verifying OTPs

This section outlines the API endpoints available for sending and verifying SMS OTPs. The service supports both **manual** and **automatic** OTP generation to meet diverse client requirements.

Customizing SMS Text

To send OTP messages with a custom sender ID and customized SMS text, follow the steps below:

1. **DLT Template Registration**
   Register your SMS content template on any telecom operator's DLT portal. Ensure the content complies with regulatory requirements and includes placeholders where applicable.
2. **Template Approval and Submission**
   Once the SMS content template is approved:
   a. Log in to the 2Factor Dashboard.
   b. Navigate to **SMS OTP > OTP Templates** and submit the approved template details, including the DLT registration information.
3. **Need Support?**
   For guidance with the template registration or submission process, contact us at **support@2factor.in**.

This approach ensures compliance with regulatory standards while allowing businesses to customize their SMS OTP messages.

Send OTP - AUTOGEN

The following endpoints are designed for scenarios where **2Factor.in automatically generates a random OTP** and sends it to the end user.

Sending OTP

Use the specified API endpoint to send a randomly generated OTP to the end user. This eliminates the need for manual OTP generation on the client side.

Verifying OTP

To verify the OTP value entered by the end user, utilize the **VERIFY OTP** API endpoints. These endpoints ensure the accuracy and validity of the entered OTP against the generated value.

These endpoints streamline the OTP management process, providing secure and efficient functionality for user authentication workflows.

GET Send OTP (AUTOGEN)

```
https://2factor.in/API/V1/:api_key/SMS/:phone_number/AUTOGEN/:otp_template_name
```

Use this endpoint to send a system-generated 6-digit OTP directly to the end user. The OTP is automatically generated by **2Factor.in**, ensuring randomness and security.

PATH VARIABLES

api_key	XXXX-XXXX-XXXX-XXXX-XXXX APIKey value obtained from 2Factor.in
phone_number	+919999999999 Phone number formated in the international number format (+
otp_template_name	OTP1 (*optional) Name of the OTP template to be sent. In absense of any value, a default OTP template would be picked up.

GET Send OTP (AUTOGEN2)

```
https://2factor.in/API/V1/:api_key/SMS/:phone_number/AUTOGEN2/:otp_template_nam
e
```

Use this endpoint to send a system-generated 6-digit OTP to the end user. The generated OTP value is included in the **API response**, allowing your application to directly use it for validation purposes.

Sample API Response

Plain Text

```
jsonCopy code{
  "Status": "Success",
  "Details": "l60gaseavbemjvlgw7o-7303503698702e11",
  "OTP": "565059"
}
```

- **Status**: Indicates the success of the API request.
- **Details**: A unique identifier for the OTP request.
- **OTP**: The 6-digit OTP generated by the system.

Verifying OTP

To verify the OTP entered by the user, you have two options:

1. Use the **VERIFY** endpoint for server-side validation.
2. Match the OTP entered by the user with the `OTP` value provided in the API response.

PATH VARIABLES

api_key	XXXX-XXXX-XXXX-XXXX-XXXX APIKey value obtained from 2Factor.in
phone_number	+919999999999

Phone number formated in the international number format (
+

otp_template_name	OTP1 (*optional) Name of the OTP template to be sent. In absense of any value, a default OTP template would be picked up.

GET Send OTP (AUTOGEN3)

```
https://2factor.in/API/V1/:api_key/SMS/:phone_number/AUTOGEN3/:otp_template_nam
e
```

Sending OTP

Use this endpoint to send a system-generated 4-digit OTP to the end user. The OTP is automatically generated and securely transmitted.

Verifying OTP

To validate the OTP entered by the user, use the **VERIFY** endpoint provided below. This ensures the entered OTP matches the system-generated value.

PATH VARIABLES

api_key	XXXX-XXXX-XXXX-XXXX-XXXX APIKey value obtained from 2Factor.in
phone_number	+919999999999 Phone number formated in the international number format (+

otp_template_name	OTP1 (*optional) Name of the OTP template to be sent. In absense of any value, a default OTP template would be picked up.

Send OTP - Custom OTP

The below endpoints are useful when you want to pass the custom generated OTP value to the end user.

GET Send OTP (Manual Generation)

```
https://2factor.in/API/V1/:api_key/SMS/:phone_number/:otp_value/:otp_template_name
```

Use this endpoint to send a **custom 4-6 digit OTP value** to the end user. This allows flexibility to define OTP values based on specific application requirements.

PATH VARIABLES

api_key	XXXX-XXXX-XXXX-XXXX-XXXX APIKey value obtained from 2Factor.in
phone_number	+919999999999 Phone number formated in the international number format (+
otp_value	12345 4-6 character OTP value to be sent to the user
otp_template_name	OTP1

	(*optional) Name of the OTP template to be sent. In absense of any value, a default OTP template would be picked up.

Verify OTP Input

2Factor provides two methods to verify the OTP entered by the user, offering flexibility based on your application requirements:

1. **Using Unique SMS Request ID + OTP Value**
   a. Verify the OTP by combining the unique request ID (provided in the API response when the OTP was sent) with the OTP value entered by the user.
   b. This method ensures precise mapping of the OTP to the specific request.
2. **Using User's Phone Number + OTP Value**
   a. Validate the OTP directly against the user's phone number and the OTP value entered by the user.
   b. This method is simpler and does not require storing the unique request ID.

These verification options ensure secure and efficient validation workflows tailored to different use cases.

GET Verify OTP (Using OTP Session Id + OTP Value)

```
https://2factor.in/API/V1/:api_key/SMS/VERIFY/:otp_session_id/:otp_entered_by_user
```

Use this endpoint to verify the OTP value entered by user.

PATH VARIABLES

api_key	XXXX-XXXX-XXXX-XXXX-XXXX

	XXXX-XXXX-XXXX-XXXX-XXXX APIKey value obtained from 2Factor.in
otp_session_id	XXXX-XXXX-XXXX-XXXX-XXXX OTP session id printed in the send OTP request
otp_entered_by_user	123455 OTP value entered by the end user

GET Verify OTP (Using PhoneNumber + OTP Value)

```
https://2factor.in/API/V1/:api_key/SMS/VERIFY3/:phone_number/:otp_entered_by_user
```

Use this endpoint to verify the OTP value entered by user.

PATH VARIABLES

api_key	XXXX-XXXX-XXXX-XXXX-XXXX APIKey value obtained from 2Factor.in
phone_number	91XXXXXXXXXX Customer's phone number
otp_entered_by_user	123455 OTP value entered by the end user

Sandbox / Testing

2Factor.in Sandbox Environment

Overview

The 2Factor.in Sandbox Environment allows developers to integrate and test SMS OTP APIs without sending actual SMS messages or incurring charges. This testing environment provides realistic API responses that mimic production behavior, enabling seamless development and testing workflows.

How to Enable Sandbox Mode

To enable sandbox/testing endpoints for your account:

1. **Send an email to**: `support@2factor.in`
2. **Include the following information**:
   a. Your account API key
   b. Specific sample phone number(s) for testing (e.g., `+919920123456`)
   c. Request to enable sandbox mode for testing purposes
3. **Wait for confirmation**: Our support team will confirm when sandbox mode is activated for your account and specified phone numbers.

Sandbox Endpoints

Once enabled, all API endpoints will return dummy responses instead of sending actual SMS messages. The sandbox maintains the same URL structure as production endpoints.

Base URL

Plain Text

```
https://2factor.in/API/V1/{your-api-key}/SMS/
```

SMS Generation Endpoints

1. Auto-Generated OTP (AUTOGEN)

Endpoint Pattern:

Plain Text

```
GET https://2factor.in/API/V1/{api-key}/SMS/{phone}/AUTOGEN/{template}
GET https://2factor.in/API/V1/{api-key}/SMS/{phone}/AUTOGEN
```

Example Request:

Plain Text

```
GET https://2factor.in/API/V1/XXXX-XXXX-XXXX-XXXX-XXXX/SMS/+919920123456/
```

Sandbox Response:

json

```json
{
  "Status": "Success",
  "Details": "mf2djpsmqnrwmwob7z-978c24505cf48a13"
}
```

Response Fields:

- `Status`: Always "Success" in sandbox
- `Details`: Unique session ID for OTP verification (randomly generated)

---

2. Auto-Generated OTP with OTP in Response (AUTOGEN2)

Endpoint Pattern:

Plain Text

```
GET https://2factor.in/API/V1/{api-key}/SMS/{phone}/AUTOGEN2/{template}
GET https://2factor.in/API/V1/{api-key}/SMS/{phone}/AUTOGEN2
```

Example Request:

Plain Text

```
GET https://2factor.in/API/V1/XXXX-XXXX-XXXX-XXXX-XXXX/SMS/+919920123456/
```

Sandbox Response:

json

```json
{
  "Status": "Success",
  "Details": "mf2dxd0bhngrkfjm79j-978c33db288487d8",
  "OTP": "521983"
}
```

Response Fields:

- `Status` : Always "Success" in sandbox
- `Details` : Unique session ID for OTP verification
- `OTP` : 6-digit OTP code (randomly generated)

---

3. Custom Template SMS

Endpoint Pattern:

Plain Text

```
GET https://2factor.in/API/V1/{api-key}/SMS/{phone}/{template-name}
```

Example Request:

Plain Text

```
GET https://2factor.in/API/V1/XXXX-XXXX-XXXX-XXXX-XXXX/SMS/+919920123456/
```

Sandbox Response:

json

```json
{
  "Status": "Success",
  "Details": "mf2djpsmqnrwmwob7z-978c24505cf48a13"
}
```

OTP Verification Endpoints

1. Standard OTP Verification (VERIFY)

Plain Text

```
GET https://2factor.in/API/V1/{api-key}/SMS/VERIFY/{session-id}/{otp}
```

Example Request:

Plain Text

```
GET https://2factor.in/API/V1/XXXX-XXXX-XXXX-XXXX-XXXX/SMS/VERIFY/mf2ddmr
```

Sandbox Response (Default):

json

```json
{
  "Status": "Error",
  "Details": "OTP Mismatch"
}
```

Special Test Case:
For testing successful verification, use the special test session and OTP:

Plain Text

```
GET https://2factor.in/API/V1/{api-key}/SMS/VERIFY/Dummy-12345/12345
```

Success Response:

json

```json
{
  "Status": "Success",
  "Details": "OTP Matched"
}
```

---

2. Alternative OTP Verification (VERIFY2)

Endpoint Pattern:

Plain Text

```
GET https://2factor.in/API/V1/{api-key}/SMS/VERIFY2/{session-id}/{otp}
```

Example Request:

Plain Text

```
GET https://2factor.in/API/V1/XXXX-XXXX-XXXX-XXXX-XXXX/SMS/VERIFY2/mf2ddm
```

Sandbox Response:

json

```json
{
  "Status": "Error",
  "Details": "OTP Mismatch"
}
```

Note: VERIFY2 endpoints always return "OTP Mismatch" in sandbox mode for testing error handling.

Testing Workflow Example

Here's a complete testing workflow using sandbox endpoints:

Step 1: Generate OTP

bash

```bash
curl "https://2factor.in/API/V1/your-api-key/SMS/+919920123456/AUTOGEN2/T
```

Response:

json

```json
{
  "Status": "Success",
  "Details": "mf2dxd0bhngrkfjm79j-978c33db288487d8",
  "OTP": "521983"
}
```

Step 2: Verify OTP (Testing Mismatch)

bash

```bash
curl "https://2factor.in/API/V1/your-api-key/SMS/VERIFY/mf2dxd0bhngrkfjm7
```

Response:

json

```json
{
  "Status": "Error",
  "Details": "OTP Mismatch"
}
```

Step 3: Test Success Case

```bash
curl "https://2factor.in/API/V1/your-api-key/SMS/VERIFY/Dummy-12345/12345
```

Response:

```json
{
  "Status": "Success",
  "Details": "OTP Matched"
}
```

Important Notes

⚠️ Sandbox Limitations

- **No actual SMS sent**: All endpoints return dummy responses
- **Random session IDs**: Each request generates a new random session ID
- **Fixed test credentials**: Only `Dummy-12345/12345` returns success for VERIFY
- **Account-specific**: Sandbox mode must be enabled per account and phone number

🔧 Development Tips

1. **Use AUTOGEN2** for testing complete workflows (includes OTP in response)
2. **Test both success and failure scenarios** using appropriate endpoints
3. **Session IDs are random** - don't hardcode them in your tests
4. **Always test error handling** using VERIFY2 endpoints

📞 Support

- **Email**: support@2factor.in
- **Subject**: "Sandbox Environment Setup Request"
- **Include**: API key, test phone numbers, and use case description

---

HTTP Status Codes

All sandbox endpoints return HTTP 200 with JSON response body. Check the `Status` field in the response to determine success or failure:

- `"Status": "Success"` - Operation completed successfully
- `"Status": "Error"` - Operation failed (check `Details` for reason)

Rate Limits

Sandbox endpoints do not follow any rate limiting as production endpoints.

Send Transactional SMS

Sending Non-Promotional SMS

Use the following endpoint to send **Non-Promotional SMS** requests categorized under **Service Implicit**. This endpoint allows delivery of SMS text approved for **Service Implicit** or **Service Explicit** message types on the DLT platform.

Important Notes

- **Compliance Requirement**: Effective **April 1, 2021**, only DLT-approved SMS text templates are allowed via 2Factor. Ensure your SMS templates are registered and approved on the DLT portal before initiating requests.

This endpoint ensures compliance with regulatory standards while enabling the secure delivery of service-based SMS messages.

Send Single SMS

POST Send Single SMS

```
https://2factor.in/API/R1/
```

Sending Single Transactional SMS

Use this endpoint to send **Transactional SMS**, including:

- OTPs
- Order notifications
- **Service Implicit** and **Service Explicit** messages

Important Notes

- **Compliance Requirement**: Starting **April 1, 2021**, only SMS formats approved through the DLT platform are supported. Ensure all message templates are registered and approved on the DLT portal prior to sending.

Body urlencoded

module	TRANS_SMS Module name - TRANS_SMS for transactional SMS
apikey	7e825d24-XXXX-XXXX-XXXX-0200cd93604211 API key issued against your account

to	91XXXXXXXXXX,91YYYYYYYYYY Phone numbers separated by comma
from	HEADER DLT approved sender id
msg	DLT Approved Message Text Goes Here DLT approved SMS text
scheduletime	2022-01-01 13:27:00 (*optional) Time at which SMS needs to be triggered Note: Schedule time must be at least 5 minutes greater than the current time
peid	DLT Registration Number (*optional) PE Id of the DLT approved content template
ctid	DLT Content Template Id (*optional) CT Id of the DLT approved content template
campaignname	Name For a Click Tracking Campaign (*optional) To enable dynamic shortlinking of a URL in the message & track clicks
campaignwebhook	URL for receiving Webhook notification on shortlink clicks (*Optional) Publicly accessible URL for receiving realtime webhook about link click

Send Bulk Messages

The 2Factor Bulk SMS API allows you to send multiple SMS messages in a single API request. This is ideal for sending transactional or promotional messages to multiple recipients simultaneously.

POST Send Bulk SMS

```
https://2factor.in/API/R1/Bulk/
```

Sending Bulk SMS

This endpoint allows you to send Bulk SMS messages using the 2Factor.in API.

Endpoint

- **URL**: `https://2factor.in/API/R1/Bulk/`
- **Method**: `POST`
- **Headers**:
    - `Content-Type` : `application/json`

Request Details

Input Fields and Their Meanings

Field	Type	Required	Description
`module`	`string`	Yes	Specifies the type of operation. Must always be set to `"TRANS_SMS"`.
`apikey`	`string`	Yes	Your unique API key for authentication. Ensure it's valid and linked to an active service.
`messages`	`array`	Yes	An array of SMS message objects. Each object represents an individual SMS to be sent.

Message Object (Each Item in the `messages` Array)

	Type	Required	Description	Example Value
smsFrom	string	Yes	DLT-approved Sender ID (6 characters max)	TFACTR
smsTo	string	Yes	Recipient mobile with country code	+919999999999
smsText	string	Yes	DLT approved message content	Your OTP is 123456
callback_value1	string	No	Custom value echoed back in webhook	order_12345
callback_value2	string	No	Second custom value echoed back	user_789

Sample Request Payload

Plain Text

```
{
  "module": "TRANS_SMS",
  "apikey": "XXXXX-XXXX-XXXX-XXXX-XXXX",
  "messages": [
    {
      "smsFrom": "TFACTR",
      "smsTo": "+919876543210",
      "smsText": "Welcome! Your verification code is 483920",
      "callback_value1": "order_98765",
        "callback_value2": "cust_1122",
```

Response Details

Successful Response

Field	Type	Description
`success`	`boolean`	Indicates whether the request was successfully processed.
`requestId`	`string`	A unique identifier for tracking the request.
`messageCount`	`integer`	The total number of messages processed in the request.
`timestamp`	`string`	The time at which the request was processed, in ISO 8601 format.
`processingTime`	`integer`	The total time (in milliseconds) taken to process the request.
`queueStats`	`object`	Details about the queue processing, including successful and failed chunks.
`messages`	`array`	Contains details of each

Example Successful Response

API Response prints requests which are successfully accepted, as well as requests which have been rejected for input validations on the phone number value.

Plain Text

```
{
  "success": true,
  "requestId": "f4bda732-XXXX-XXXX-XXXX-518b4ea82708",
  "messageCount": 2,
  "timestamp": "2024-12-16T10:59:17.311Z",
  "processingTime": 1224,
  "queueStats": {
    "success": true,
    "totalChunks": 1,
    "successfulChunks": 1,
    "failedChunks": 0
```

Error Responses

Field	Type	Description
`success`	`boolean`	Indicates whether the request was successfully processed. Always `false` for errors.
`requestId`	`string`	A unique identifier for tracking the request.
`error`	`string`	A short description of the error encountered.
`errorType`	`string`	The category of error (e.g., `ValidationError`, `UnexpectedError`).
`details`	`array`	Detailed information about the validation or processing errors.
`timestamp`	`string`	The time at which the error occurred, in ISO 8601 format.

Example Error Responses

Case 1: Invalid API Key

Plain Text

```
{
  "success": false,
  "requestId": "e3de200f-XXXX-XXXX-XXXX-2a9255bb7c51",
  "error": "Validation failed",
  "errorType": "ValidationError",
  "details": [
    {
      "field": "apikey",
      "error": "Invalid API Key Or TRANSACTIONAL_SMS Service Is Inactive"
    }
  ],
```

Case 2: Invalid JSON Format

Plain Text

```
jsonCopy code{
  "success": false,
  "requestId": "147dbbef-XXXX-XXXX-XXXX-6271da57cdca",
  "error": "Validation failed",
  "errorType": "ValidationError",
  "details": [
    {
      "error": "Failed to parse request body as JSON",
      "details": "Expected ',' or ']' after array element in JSON at posi
    }
  ],
```

Webhook / Callback Payload (sent to your URL)

JSON

Plain Text

```
{
  "SessionId": "30b3c545-5e1a-482e-9fac-1a76e805020d-1",
  "SmsFrom": "TFACTR",
  "SmsTo": "9876543210",
  "SmsStatus": "DELIVERED",
  "SentAt": "2025-11-24 19:43:06",
  "StatusAt": "2025-11-24 19:43:08",
  "callback_value1": "order_98765",
  "callback_value2": "cust_1122"
```

Key Validation Rules

1. **Module**: Must always be `"TRANS_SMS"`.
2. **API Key**: Must be valid and linked to an active service.
3. **Messages Array**: Must contain at least one message and not exceed 1000 messages per request.
4. **Phone Numbers**:
    a. Must be in international format (e.g., `+91XXXXXXXXXX`).
    b. Only numeric characters are allowed, with an optional leading `+`.
5. **SMS Content**: Message length must not exceed 1000 characters.

Notes

- Double-check your JSON structure to avoid formatting errors.
- Use authorized Sender IDs (`smsFrom`) configured in your account.
- Validate phone numbers and SMS text before sending the request to reduce errors.

HEADERS

Content-Type	application/json

Body raw (json)

json

```json
{
    "module": "TRANS_SMS",
    "apikey": "XXXX-XXXX-XXXX-XXXX-XXXX",
    "messages": [
        {
            "smsFrom": "HEADER",
            "smsTo": "+91XXXXXXXXXX",
            "smsText": "This is your first message.",
            "callback_value1": "Enter Custom value",
            "callback_value2": "Enter Custom value",
            "callback_format": "json"
        },
        {
            "smsFrom": "HEADER",
            "smsTo": "+91XXXXXXXXXX",
            "smsText": "This is your second message.",
            "callback_value1": "Enter Custom value",
            "callback_value2": "Enter Custom value",
            "callback_format": "json"
        }
    ]
}
```

Send Promotional SMS

Sending Promotional SMS

Use this endpoint to send **Promotional SMS** to your clients, including:

- Offers and discounts
- Seasonal greetings
- Coupon codes
- Re-targeting messages

Important Notes

- **Message Types**: Supports **Service Explicit** and **Promotional Messages** as approved on the DLT platform.
- **Compliance Requirement**: Starting **April 1, 2021**, only DLT-approved SMS formats are allowed. Ensure all promotional content is registered and approved on the DLT portal before initiating requests.

POST Send SMS

```
https://2factor.in/API/R1/
```

Purpose

This endpoint is used to send **Promotional SMS**, such as:

- Offers and discounts
- Seasonal greetings
- Coupon codes
- Re-targeting messages

Message Categories

Supports **Service Explicit** and **Promotional Messages** as registered on the DLT platform.

Body urlencoded

module	PROMO_SMS Module name - PROMO_SMS for Promotional SMS
apikey	7e825d24-XXXX-XXXX-XXXX-0200cd93604211 API key issued against your account

to	91XXXXXXXXXX,91YYYYYYYYYY Phone numbers separated by comma
from	HEADER DLT approved sender id
msg	DLT Approved Message Text Goes Here DLT approved SMS text
scheduletime	2022-01-01 13:27:00 (*optional) Time at which SMS needs to be triggered Note: Schedule time must be at least 5 minutes greater than the current time

Encrypted SMS Endpoints (*Premium)

The **Encrypted SMS API** provides secure methods to send SMS messages and retrieve their delivery statuses. The API ensures data confidentiality using **AES-256 bit encryption in GCM & CBC mode.__**

Authentication

All API requests require the **Secret-Key** in the headers for authentication. The key must be encrypted using **AES-256 GCM / CBC mode**.

Endpoints

1. Send SMS

Endpoint Information

- **URL**: `https://2factor.in/API/R1?module=ENC&encmode=GCM`

- **Method:** `POST`
- **Headers**:
  - `Secret-Key`: The AES-256 bit encrypted key.

Request Payload

The request requires an encrypted JSON payload structured as follows:

```c
{
    "module": "TRANS_SMS",
    "apikey": "954817d9-XXXX-XXXX-XXXX-XXXX-cd936042",
    "to": "91XXXXXXXXXX",
    "from": "SENDER",
    "msg": "Your DLT Approved SMS Text"
}
```

Steps to Encrypt and Send

1. **Prepare JSON Payload**: Use the format above with valid API key and message details.
2. **Encrypt the Payload**: Encrypt using AES-256 CBC / GCM mode with the provided sandbox secret key.
3. **Send Encrypted Payload**: Include the encrypted value in the body of the POST request. `encmode supported CBC & GCM values to indicate the mode of encryption. default encryption mode is CBC`

Sample Request (cURL)

```Plain Text
curl -X POST "https://2factor.in/API/R1/?module=ENC" \
-H "Secret-Key: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX" \
-d "<EncryptedPayload>"
```

---

2. Retrieve SMS Delivery Status

Endpoint Information

- **URL**: `https://2factor.in/API/R1?module=ENC`
- **Method**: `POST`
- **Headers**:
  - `Secret-Key` : The AES-256 bit encrypted key.

Request Payload

The request requires an encrypted JSON payload structured as follows:

Plain Text

```
{
    "module": "TRANS_SMS_DLR",
    "apikey": "54817d9-XXXX-XXXX-XXXX-XXXX-cd936042",
    "sessionid": "XXXXXXXXXXXXX-8477e7fc9c7749c6",
    "smsdate": "2024-01-18"
}
```

Steps to Encrypt and Send

1. **Prepare JSON Payload**: Use the format above with valid API key, session ID, and date.
2. **Encrypt the Payload**: Encrypt using AES-256 GCM mode with the provided sandbox secret key.
3. **Send Encrypted Payload**: Include the encrypted value in the body of the POST request.

Sample Request (cURL)

Plain Text

```
curl -X POST "https://2factor.in/API/R1/?module=ENC" \
-H "Secret-Key: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX" \
-d "<EncryptedPayload>"
```

Responses

General Response Structure

All API responses are in JSON format. Below are common scenarios and their respective responses:

Note: Error responses are printed in json plaintext , **Success** respomses would be in an encrypted format as given in the below examples

Scenario	HTTP Code	Response Example
Successful request	200	`{"Status": "Success", "Details": ""}`
Invalid or missing `Secret-Key`	400	`{"Status": "Error", "Details": "Secret-Key is missing in the request header"}`
Invalid payload encryption	400	`{"Status": "Error", "Details": "Decryption failed..."}`
GET method used instead of POST	400	`{"Status": "Error", "Details": "HTTP GET is not supported on this endpoint"}`

Example Decrypted Responses

For Sending SMS

Plain Text

```
{
    "Status": "Success",
    "Details": "wYW/rnHQquWW2U7d0fk9PQniy5oGI+fbkFazRtrY4FEfCzt241yqf2jPe
}
```

The decrypted `Details` field provides a unique SMS request ID.

For Retrieving Delivery Status

Plain Text

```
{
    "Status": "Success",
    "Details": "1 record found for the specified filters",
    "DeliveryStatus": "DELIVERED",
    "SentAt": "2024-01-18 19:25:41",
    "DeliveredAt": "2024-01-18 19:42:33"
}
```

Or, if no records are found:

Plain Text

```
{
    "Status": "Success",
    "Details": "No records found for the specified filters"
}
```

Error Handling

The API provides clear error messages for troubleshooting. Common errors include:

Error Condition	HTTP Code	Description
Missing `Secret-Key` header	400	The `Secret-Key` header is required for authentication.
Invalid `Secret-Key` value	400	Ensure the correct encryption key is provided.
Payload not properly formatted or encoded	400	The payload must be encrypted and Base64-encoded before submission.
Incorrect HTTP method used	400	The endpoint supports only POST requests.

Security Best Practices

- **Encryption**: Ensure all payloads are encrypted with AES-256 GCM mode.
- **Transport Layer Security**: Always use HTTPS for secure transmission.
- **Key Management**: Safeguard your `Secret-Key` and `API-Key` and avoid exposing them in public repositories or documentation.
- **Response Validation**: Decrypt and validate responses to ensure the integrity of the data.

Read More

Dynamic Shortlinking in Transactional SMS API (for CTA Whitelisting)

In alignment with the latest **TRAI guidelines**, all URLs included in SMS text must be pre-whitelisted on the DLT platform. To meet this compliance requirement, **2Factor** introduces the **Dynamic Shortlinking Feature** as part of the Transactional SMS API.

Feature Highlights

1. **Dynamic URL Shortening**
    a. **Automatic URL Shortening**: URLs in SMS messages are dynamically shortened at runtime.
    b. **DLT Compliance**: Shortened URLs adhere to DLT-approved formatting standards.
        i. **Example**: `https://e.1rp.in/DLT_HEADER/h6fgsf`
        ii. For example, if your SMS header is `TFACTR`, URLs in your SMS will be replaced with shortened versions like:
            a. `https://e.1rp.in/TFACTR/h6fgs`
2. **Free Alternative to Third-Party Services**: This feature serves as a free replacement for services like Bit.ly.

How to Enable URL Shortening

1. Add the parameter `shortenurl=1` in your **R1 - Transactional API** request to activate the feature.
2. Whitelist the following URL format on the **DLT portal under CTA-Whitelisting**:
    a. `https://e.1rp.in/DLT_HEADER/\\*`
3. This ensures all dynamically created short URLs comply with DLT regulations.

Webhook Support for Click Tracking

- **Real-Time Tracking**: Monitor user interaction with shortened URLs in real-time.
- **Enable Click Tracking**:
    - Use the parameter `shortenurlwebhook` to specify the webhook URL for click notifications.
    - Pass a custom callback value using the parameter `shortenurlcallback` to include it in webhook payloads.

Runtime PEID and CTID Support in Transactional SMS API

To eliminate delays caused by DLT template replication (previously up to 4 hours), clients can now pass **DLT Principal Entity ID (PEID)** and **DLT Content Template ID (CTID)** in real-time during the API request.

Sample API Request

Plain Text

```
plaintextCopy codehttps://2factor.in/API/R1/?module=TRANS_SMS&apikey=API_
```

Parameter Notes

- `PE_ID` : Replace with the Principal Entity ID of the DLT-approved message.
- `CT_ID` : Replace with the Content Template ID of the DLT-approved message.

URL Shortening and Click Tracking (for Campaign Performance Tracking)

The Transactional SMS API now includes URL shortening with **real-time click tracking**, enabling businesses to measure campaign performance effectively.

Key Benefits

1. Dynamically shorten URLs included in SMS messages.
2. Monitor user interactions, such as phone numbers that access the links.
3. Receive real-time click data via webhook.

Sample API Request

Plain Text

```
plaintextCopy codehttps://2factor.in/API/R1/?module=TRANS_SMS&apikey=API_
```

Parameter Notes

- `campaignwebhook` **(Optional)**: A URL-encoded callback URL to receive real-time click notifications.
    - If omitted, URL shortening is enabled, but click tracking will not be activated.

Webhook Callback Details

When a user clicks on a shortened link, **2Factor** sends an HTTP POST request to the specified `campaignwebhook` URL. The payload contains the following information:

Field	Description
`accessed`	Timestamp when the link was accessed (format: `YYYY-MM-DD HH:MM:SS`).
`phoneNo`	The phone number of the user who clicked the link.
`campaignName`	The campaign name associated with the message.
`ip`	IP address of the user who clicked the link.

Example Webhook Payload

Plain Text

```
plaintextCopy codeaccessed=2024-11-29 14:30:00&phoneNo=9876543210&campaig
```

Developer Guidelines

1. **Parameter Encoding**: Ensure all parameters are properly URL-encoded before making API requests.
2. **API Security**:
    a. Use IP-based restrictions to secure API access.
    b. Regularly rotate API keys to prevent unauthorized access.

3. For support, contact **2Factor Support**.

WhatsApp Messaging APIs

For pricing and setup, see our WhatsApp Business API Pricing and WhatsApp Business API Integration Guide.

Whatsapp Voice Calling APIs

We also provide a dedicated WhatsApp Voice Calling API for businesses looking to add call capabilities.

Learn More About WhatsApp Solutions

- WhatsApp Chatbot Provider
- WhatsApp Business API Guide 2025
- WhatsApp Business API Pricing
- WhatsApp Voice Calling API
- WhatsApp API Service Provider in India
- Official WhatsApp API Provider
- WhatsApp Business API Integration