



Postman guide for messaging APIs

How to use Postman to explore and test a messaging API: environments, authentication, sending a message, checking status and testing delivery-report webhooks.

Educational guide. Educational and proposed examples only. Wise Network does not operate a messaging API: the address `api.wisenetwork.co` and every request in this guide and in the example collection are placeholders that will not work.

Online version: <https://wisenetwork.co/developers/postman/> · Content updated 29 October 2026

What Postman is for

Postman is a desktop and web application for sending HTTP requests and inspecting responses without writing code. Teams use it to learn an API, to share working examples as collections, and to run simple automated checks. It is useful during an integration project, before or alongside writing your own client code.

The example collection

We provide an example collection that shows the structure described on this page. It contains placeholder variables and four requests: get an access token, send a message, check its status, and post a sample delivery report to your own test server.

[Download the example collection \(JSON\)](#)

The collection is **educational**. Its base address `https://api.wisenetwork.co/v1` is a placeholder: there is no Wise Network API behind it, so its requests return errors. Use it as a template for a provider whose API is live, or to agree the design of a future interface.

Importing it

- 1. Import.** In Postman choose Import and select the downloaded file.
- 2. Review the variables.** Open the collection, then the Variables tab.
- 3. Replace placeholders.** Set real values only for a provider that has given you an API and credentials.

Environments and variables

Variables keep addresses and secrets out of individual requests. Write them in double curly braces, such as `{{baseUrl}}`, and Postman substitutes the value when it sends the request. An environment is a named set of values, for example one for testing and one for production.

Variables used in the example collection

Variable	Purpose	Example value
baseUrl	Root address of the API	<code>https://api.wisenetwork.co/v1</code> (placeholder)
clientId, clientSecret	Credentials exchanged for a token	Issued by the provider; keep secret
accessToken	Short-lived token set by the token request	Filled in automatically
sender	Approved sender ID	ExampleCo
testNumber	Destination for tests	+12025550143 (fictional)
messageId	Stored from the send response	Filled in automatically
webhookUrl	Your own HTTPS endpoint for receipts	<code>https://your-app.example.com/hooks/sms</code>

Mark credentials as secret variables, keep real values in the current value field (which is not synced or shared), and never export or share an environment that contains live credentials.

Authentication

Set authentication once on the collection so every request inherits it. The example collection uses a bearer token obtained with client credentials; other providers use an API key header or HTTP Basic authentication. See [messaging API concepts](#) for the trade-offs.

Proposed example: token request. Not a Wise Network endpoint.

```
POST {{baseUrl}}/oauth/token
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials&client_id={{clientId}}&client_secret={{clientSecret}}
```

A test script on that request can save the token for the other requests:

Postman test script (JavaScript), illustrative

```
const body = pm.response.json();
pm.test("token issued", () => pm.expect(body.access_token).to.be.a("string"));
pm.collectionVariables.set("accessToken", body.access_token);
```

Sending a message

Proposed example: send request. Not a Wise Network endpoint.

```
POST {{baseUrl}}/messages
Authorization: Bearer {{accessToken}}
Content-Type: application/json
Idempotency-Key: {{$guid}}
```

```
{
  "from": "{{sender}}",
  "to": ["{{testNumber}}"],
  "text": "Test message from Postman",
  "dlr_url": "{{webhookUrl}}"
}
```

{{\$guid}} is a Postman dynamic variable that creates a new unique value for each request, so a repeated request is not treated as a new message. Store the returned identifier for the next step:

Postman test script (JavaScript), illustrative

```
pm.test("accepted", () => pm.expect(pm.response.code).to.be.oneOf([200, 201, 202]));
const id = pm.response.json().messages?.[0]?.id;
pm.collectionVariables.set("messageId", id);
```

Checking status

Proposed example: status request. Not a Wise Network endpoint.

```
GET {{baseUrl}}/messages/{{messageId}}
Authorization: Bearer {{accessToken}}
```

The status moves from accepted towards a final state such as delivered, undeliverable or expired. Do not poll in a tight loop; delivery reports are the efficient way to receive final states. The meaning of each state is explained in [delivery reports and errors](#).

Testing delivery-report webhooks

A webhook is a request the provider sends to your server. You cannot receive it inside Postman, but you can use Postman to send a sample receipt to your own test endpoint and check that your code verifies the signature, stores the result and answers quickly with a 2xx status.

Sample delivery report sent to your own server. Format is illustrative.

```
POST {{webhookUrl}}
Content-Type: application/json
X-Signature: t=1793197860,v1=<hex-hmac>

{ "id": "{{messageId}}", "status": "delivered", "done_at": "2026-10-28T14:31:00Z" }
```

Also test the failure cases: a wrong signature, an old timestamp, the same report sent twice, and a report for an unknown message ID.

Automated checks with the Collection Runner

The Collection Runner sends the requests in order and reports the result of each test script. Use it with a test environment to repeat a full scenario (token, send, status) after a change. Keep runs against live services within the test volume agreed with the provider.

Good practice

- Use a separate environment for testing, and switch to production deliberately.
- Keep secrets in secret variables and current values only; review a collection before sharing it.
- Send test messages only to numbers you control or that the provider has agreed.
- Record message IDs and timestamps from test runs; they are what support teams need. See [integration checklist](#).

Planning an integration with Wise Network? [Contact our messaging team](#) about the connection options available today.

Questions? Email support@wisenetnetwork.co. This PDF is generated from the web page above; the online version is the reference.