

@apiclient.xyz/mailgun un

Documentation for @apiclient.xyz/mailgun

- [readme.md for @apiclient.xyz/mailgun](#)
- [changelog.md for @apiclient.xyz/mailgun](#)

readme.md for @apiclient.xyz/mailgun

“ A modern, TypeScript-first API wrapper for Mailgun with smart fallback mechanisms

Send transactional emails through Mailgun's powerful API with an elegant, type-safe interface. This package seamlessly integrates with the [@push.rocks/smartmail](https://push.rocks/smartmail) ecosystem and provides automatic SMTP fallback for edge cases.

Why @apiclient.xyz/mailgun?

-  **TypeScript Native** - Full type safety with excellent IntelliSense support
-  **Smart Fallback** - Automatically falls back to SMTP when API limitations are hit
-  **Attachment Support** - Hassle-free file attachments with built-in handling
-  **Multi-Region** - Support for both EU and US Mailgun regions
-  **Message Retrieval** - Fetch stored messages from Mailgun's API
-  **Webhook Integration** - Easy integration with Mailgun webhooks
-  **Modern API** - Clean, fluent interface powered by smartrequest v4

Installation

```
npm install @apiclient.xyz/mailgun
# or
pnpm install @apiclient.xyz/mailgun
# or
yarn add @apiclient.xyz/mailgun
```

Quick Start

```
import { MailgunAccount } from '@apiclient.xyz/mailgun';
import { Smartmail } from '@push.rocks/smartmail';

// Initialize your Mailgun account
const mailgun = new MailgunAccount({
  apiToken: 'your-mailgun-api-token',
  region: 'eu' // or 'us'
});

// Create and send an email
const email = new Smartmail({
  from: 'Your App <noreply@yourdomain.com>',
  subject: 'Welcome to Our Service! 🎉',
  body: '<h1>Hello!</h1><p>Thanks for signing up.</p>'
});

await mailgun.sendSmartMail(email, 'user@example.com');
```

Core Features

📧 Sending Emails

The primary way to send emails is through the `sendSmartMail()` method, which accepts a `Smartmail` object:

```
import { MailgunAccount } from '@apiclient.xyz/mailgun';
import { Smartmail } from '@push.rocks/smartmail';

const mailgun = new MailgunAccount({
  apiToken: process.env.MAILGUN_API_TOKEN,
  region: 'eu'
});

// Create a rich HTML email
const email = new Smartmail({
  from: 'Team <hello@yourdomain.com>',
  subject: 'Monthly Newsletter',
```

```
body: `
  <html>
    <body>
      <h1>What's New This Month</h1>
      <p>Check out our latest updates...</p>
    </body>
  </html>
`
});

// Send to a recipient
await mailgun.sendSmartMail(email, 'subscriber@example.com');
```

☐☐ Email Attachments

Add attachments seamlessly using the smartmail interface:

```
import { SmartFile } from '@push.rocks/smartfile';

const email = new Smartmail({
  from: 'Sales <sales@yourdomain.com>',
  subject: 'Your Invoice',
  body: '<p>Please find your invoice attached.</p>'
});

// Add attachment from file
const pdfFile = await SmartFile.fromFilePath('./invoice.pdf');
email.addAttachment(pdfFile);

await mailgun.sendSmartMail(email, 'customer@example.com');
```

☐☐ SMTP Fallback

For edge cases where the Mailgun API has limitations (like empty email bodies), the package automatically falls back to SMTP:

```
// Configure SMTP credentials for fallback
// Format: domain|username|password
```

```
await mailgun.addSmtpCredentials(
  'yourdomain.com|postmaster@yourdomain.com|your-smtp-password'
);

// This email with empty body will automatically use SMTP
const emptyBodyEmail = new Smartmail({
  from: 'System <system@yourdomain.com>',
  subject: 'System Notification',
  body: ''
});

await mailgun.sendSmartMail(emptyBodyEmail, 'admin@example.com');
// ☐ Automatically sent via SMTP instead of API
```

☐ Retrieving Messages

Fetch stored messages from Mailgun's API:

```
// Retrieve by message URL (from Mailgun storage)
const message = await mailgun.retrieveSmartMailFromMessageUrl(
  'https://sw.api.mailgun.net/v3/domains/yourdomain.com/messages/...'
);

if (message) {
  console.log('Subject:', message.options.subject);
  console.log('From:', message.options.from);
  console.log('Body:', message.options.body);

  // Attachments are automatically fetched
  console.log('Attachments:', message.attachments.length);
}
```

☐ Webhook Integration

Process Mailgun webhook notifications:

```
// In your webhook handler
app.post('/webhooks/mailgun', async (req, res) => {
```

```
const payload = req.body;

// Retrieve the full message from the webhook payload
const message = await mailgun.retrieveSmartMailFromNotifyPayload(payload);

if (message) {
  // Process the received email
  console.log('Received email:', message.options.subject);
}

res.status(200).send('OK');
});
```

API Reference

MailgunAccount

Constructor

```
new MailgunAccount(options: {
  apiToken: string; // Your Mailgun API token
  region: 'eu' | 'us'; // Mailgun region
})
```

Methods

`sendSmartMail(smartmail, recipient, data?)`

Send an email through Mailgun.

```
await mailgun.sendSmartMail(
  smartmailInstance,
  'recipient@example.com',
  { customData: 'optional' } // Optional template data
);
```

`addSmtplibCredentials(credentials)`

Configure SMTP fallback credentials.

```
await mailgun.addSmtpCredentials('domain|username|password');
```

```
retrieveSmartMailFromMessageUrl(url)
```

Retrieve a stored message by its Mailgun URL.

```
const message = await mailgun.retrieveSmartMailFromMessageUrl(messageUrl);
```

```
retrieveSmartMailFromNotifyPayload(payload)
```

Extract and retrieve a message from a Mailgun webhook payload.

```
const message = await mailgun.retrieveSmartMailFromNotifyPayload(webhookPayload);
```

Region Support

Mailgun operates in multiple regions. Choose the appropriate one for your needs:

- **eu** - European region (GDPR compliant, data stored in EU)
- **us** - US region (data stored in US)

```
// EU region
const mailgunEU = new MailgunAccount({
  apiToken: 'your-token',
  region: 'eu'
});

// US region
const mailgunUS = new MailgunAccount({
  apiToken: 'your-token',
  region: 'us'
});
```

Advanced Usage

Template Variables with Smartmail

Use template variables in your emails:

```
import { Smartmail } from '@push.rocks/smartmail';

const welcomeEmail = new Smartmail({
  from: 'Welcome Team <welcome@yourdomain.com>',
  subject: 'Welcome {{userName}}!',
  body: '<h1>Hi {{userName}}</h1><p>Your account is ready!</p>'
});

// Pass template data
await mailgun.sendSmartMail(
  welcomeEmail,
  'newuser@example.com',
  { userName: 'John Doe' }
);
```

Error Handling

```
try {
  await mailgun.sendSmartMail(email, 'user@example.com');
  console.log('☑ Email sent successfully');
} catch (error) {
  console.error('☒ Failed to send email:', error.message);
}
```

Batch Sending

Send multiple emails efficiently:

```
const recipients = [
  'user1@example.com',
  'user2@example.com',
  'user3@example.com'
];
```

```
const email = new Smartmail({
  from: 'Newsletter <news@yourdomain.com>',
  subject: 'Weekly Update',
  body: '<p>Here is your weekly update...</p>'
});

// Send to all recipients
await Promise.all(
  recipients.map(recipient =>
    mailgun.sendSmartMail(email, recipient)
  )
);
```

TypeScript Support

This package is written in TypeScript and provides full type definitions out of the box:

```
import {
  MailgunAccount,
  IMailgunAccountConstructorOptions,
  IMailgunMessage
} from '@apiclient.xyz/mailgun';

// Full IntelliSense support
const options: IMailgunAccountConstructorOptions = {
  apiToken: 'token',
  region: 'eu'
};

const mailgun = new MailgunAccount(options);
```

Dependencies

This package leverages the powerful @push.rocks ecosystem:

- [@push.rocks/smartrequest](#) - Modern HTTP client
- [@push.rocks/smartmail](#) - Email composition
- [@push.rocks/smartsmtplib](#) - SMTP fallback
- [@push.rocks/smartfile](#) - File handling
- [@push.rocks/smartstring](#) - String utilities

Links & Resources

- [NPM Package](#)
- [GitLab Repository](#)
- [GitHub Mirror](#)
- [TypeDoc Documentation](#)
- [Mailgun API Docs](#)

License and Legal Information

This repository contains open-source code that is licensed under the MIT License. A copy of the MIT License can be found in the [license](#) file within this repository.

Please note: The MIT License does not grant permission to use the trade names, trademarks, service marks, or product names of the project, except as required for reasonable and customary use in describing the origin of the work and reproducing the content of the NOTICE file.

Trademarks

This project is owned and maintained by Task Venture Capital GmbH. The names and logos associated with Task Venture Capital GmbH and any related products or services are trademarks of Task Venture Capital GmbH and are not included within the scope of the MIT license granted herein. Use of these trademarks must comply with Task Venture Capital GmbH's Trademark Guidelines, and any usage must be approved in writing by Task Venture Capital GmbH.

Company Information

Task Venture Capital GmbH Registered at District court Bremen HRB 35230 HB, Germany

For any legal inquiries or if you require further information, please contact us via email at hello@task.vc.

By using this repository, you acknowledge that you have read this section, agree to comply with its terms, and understand that the licensing of the code does not imply endorsement by Task Venture Capital GmbH of any derivative works.

changelog.md for @apiclient.xyz/mailgun

2025-10-17 - 2.0.3 - fix(tests)

Update testing setup and bump deps; add deno.lock and combined Node/Deno/Bun test

- Bump dev dependency @git.zone/tstest from ^2.6.1 to ^2.6.2 and dependency @push.rocks/smartrequest from ^4.3.1 to ^4.3.2 in package.json
- Add deno.lock with resolved dependency tree (new lockfile checked in)
- Introduce test/test.node+deno+bun.ts — consolidated tests for Node/Deno/Bun using Qenv and Mailgun integration flows
- Remove the old test/test.ts (replaced by the new multi-runtime test file)

2025-10-13 - 2.0.2 - fix(mailgun)

Normalize package scope and modernize Mailgun client: rename package to @apiclient.xyz/mailgun, update dependencies, refactor HTTP handling, fix types, update TS config and CI, refresh docs and tests

- Rename package and metadata to @apiclient.xyz/mailgun (package.json, npmextra.json, readme and commitinfo)
- Update dependency scopes and versions (@push.rocks packages and dev deps)
- Refactor smartrequest usage to SmartRequest fluent API, replace statusCode/body with status and async json() handling
- Adjust request/form types and attachment handling (FormField shape, filename field, SmartFile usage and Buffer conversion)
- Improve TypeScript configuration (NodeNext moduleResolution, emitDecoratorMetadata, verbatimModuleSyntax) and minor TS fixes in code
- Add Gitea CI workflows (.gitea/workflows) and pnpm workspace config
- Refresh README to reflect new package branding and usage examples
- Fix tests to await env var resolution and format calls, and update .gitignore to include AI artifacts

2022-08-07 - 2.0.0 - core

Major release with core updates.

- Core updates and fixes.
- Bumped major version to 2.0.0.

2022-08-07 - 1.0.32 - core

(BREAKING)

Breaking change: module format change.

- **BREAKING CHANGE:** switched to ECMAScript modules (ESM). Consumers may need to update import/require usage.

2022-08-07 - 1.0.31 - core

Patch release with core fixes.

- Various small core fixes and maintenance updates.

2022-08-07 - 1.0.3..1.0.30 -

patches

Series of patch releases containing routine fixes and updates.

- Multiple patch releases (1.0.3 through 1.0.30) delivering routine core fixes and minor improvements.
- No documented breaking API changes in these patch releases.

2022-08-07 - 2.0.1, 1.0.7, 1.0.2 - tags only

Version tags / metadata-only releases.

- These releases only included version tagging/packaging; no functional changes recorded.

2020-01-13 - 1.0.17 - interfaces

Export improvement.

- fix(interfaces): export interfaces so they are available to consumers.

2017-02-08 - 1.0.1 - packaging

Packaging fix.

- Fixed npmts error and packaging adjustments.

2017-02-08 - 1.0.0 - initial

Initial release.

- Initial commit ("mgun").