Mailit: OAuth 2.0 Setup Guide for Gmail and Google Workspace

Mailit Documentation

Version for Mailit 11

Contents

1 Introduction **3**

1.1 Why OAuth Authentication Is Required . . . 3

1.2 Transport Options . . . 3

2 Quick Start — End-to-End Setup **3**

2.1 OAuth for IMAP / POP / SMTP . . . 3

2.2 OAuth for Gmail REST API . . . 4

3 Settings Templates in Mailit **5**

4 Scenarios — Reference **5**

4.1 Delegated + Authorization Code . . . 5

5 Account Types — Reference **5**

6 Application Registration — Reference **6**

7 Permissions — Reference **6**

8 Mailit Connection Fields — Reference **6**

9 Server Addresses and Ports — Reference **7**

9.1 Server Addresses . . . 7

9.2 Ports . . . 7

9.3 Local Web Server and Mailit Assets . . . 7

9.3.1 Deploying Mailit Assets . . . 7

9.3.2 Available URLs . . . 8

9.3.3 Plugin Functions for the Local Server . . . 8

10 Authorization Methods — Reference **8**

10.1 Browser Login (Scenario 1) . . . 8

10.2 No Login (Scenario 2) 8

11 Token Lifecycle and Automatic Refresh **8**

11.1 Scenario 1 — Refresh Token 8

11.2 Scenario 2 — Client Credentials 9

12 Troubleshooting **9**

12.1 Diagnostic Information 9

12.2 Local Web Server Errors (-30 / -61) 9

12.3 Common Gmail OAuth Errors 9

1 Introduction

Mailit supports connecting to Gmail mailboxes using OAuth 2.0. This means Mailit does not handle your Google password. Instead, you sign in on Google's website, and they issue a temporary token to Mailit.

Google requires only one OAuth flow: **Delegated + Authorization Code**. There are no server-side Application permissions for consumer Gmail accounts.

Unlike Microsoft, Google's terminology and developer portal (Google Cloud Console) are different. This guide explains how to register the app, add scopes, and configure Mailit for both IMAP/SMTP and the Gmail REST API.

1.1 Why OAuth Authentication Is Required

> ***Important.*** *Google has disabled "Less secure app access" (Basic Authentication) for IMAP, POP, and SMTP. The only supported authentication method is OAuth 2.0. For the Gmail REST API, OAuth is also mandatory.*

1.2 Transport Options

You can connect Mailit to Gmail using two transports:

- **Classic Protocols:** IMAP (receiving), POP (receiving), SMTP (sending).
- **Gmail REST API:** A modern API provided by Google.

Both use the same OAuth flow. The only difference is the Server address and the Port.

2 Quick Start — End-to-End Setup

This section provides a complete setup for each Gmail transport. Find the one that matches your situation and follow the steps.

2.1 OAuth for IMAP / POP / SMTP

This guide explains how to connect Mailit to your Gmail mailbox using OAuth 2.0 with IMAP, POP, and SMTP. You will sign in to Google during initial setup.

Step 1: Register the application in Google Cloud Console

1. Go to the Google Cloud Console.
2. Create a new project (e.g., `Mailit OAuth`).
3. Go to **APIs & Services** → **OAuth consent screen**. Choose **External** and fill in the required fields.
4. Go to **Credentials** → **Create Credentials** → **OAuth client ID**.
5. Choose **Desktop app** as the application type.
6. Copy the **Client ID** and **Client Secret**.

Step 2: Configure OAuth in Mailit

1. Open `Simple Receiving.fmp12`.
2. Select **IMAP** (or **POP**) as the connection type and **OAuth** as the authentication method.
3. On the **OAuth 2.0 Client Credentials** tab, select the `Google Mail (Delegated)` preset.
4. Enter your **Client ID** and **Client Secret**.
5. Enter your full Gmail address in **User Name**.

Step 3: Check the receiving and sending settings

Use the following settings:

Connection	Server	Port
IMAP receiving	`imap.gmail.com`	993 (SSL/TLS)
POP receiving	`pop.gmail.com`	995 (SSL/TLS)
SMTP sending	`smtp.gmail.com`	587 (STARTTLS)

Step 4: Sign in and authorize access

Ensure the **Mailit Assets** folder is installed. Click **Open OAuth2 in External browser**. Sign in with your Google account. Because the app is unverified, you may see a warning. Click **Advanced** and **Go to Mailit (unsafe)** to proceed. Approve the requested permissions.

Step 5: Test receiving and sending

Click **Test Connection**, then **Get Mail**. To test sending, configure `Simple Sending.fmp12` using the same preset and credentials.

2.2 OAuth for Gmail REST API

This guide explains how to connect Mailit to your Gmail mailbox using the Gmail REST API.

Step 1: Register the application in Google Cloud Console

1. Go to the Google Cloud Console.
2. Create a new project (e.g., `Mailit API`).
3. Go to **APIs & Services** → **OAuth consent screen**. Choose **External**.
4. Go to **Credentials** → **Create Credentials** → **OAuth client ID**.
5. Choose **Desktop app**.
6. Copy the **Client ID** and **Client Secret**.

Step 2: Configure OAuth in Mailit

1. Open `Simple Receiving.fmp12`.
2. Select **graphRest** as the connection type and **OAuth**.
3. On the **OAuth 2.0 Client Credentials** tab, select the `Google API (Delegated)` preset.
4. Enter your **Client ID** and **Client Secret**.
5. Enter your full Gmail address in **User Name**.

Step 3: Check the receiving and sending settings

For the Gmail REST API, the server is `https://gmail.googleapis.com` and the port is 443 (HTTPS). The preset sets this automatically.

Step 4: Sign in and authorize access

Ensure the **Mailit Assets** folder is installed. Click **Open OAuth2 in External browser** and sign in with your Google account. Accept the unverified app warning and approve the permissions.

Step 5: Test receiving and sending

Click **Test Connection**, then **Get Mail**.

3 Settings Templates in Mailit

Google uses two presets in Mailit. Both use the same OAuth endpoints, but different Server addresses.

Google Mail (Delegated) — classic protocols (POP / IMAP / SMTP):

- Server: `imap.gmail.com` (for POP/SMTP, adjust manually if needed)
- SSL: `On`
- Authorization URL: `https://accounts.google.com/o/oauth2/v2/auth`
- Token URL: `https://oauth2.googleapis.com/token`
- Scope: `https://mail.google.com/`
- OAuth Add Secret To Request: `On`
- Client Secret: **required**

Google API (Delegated) — Gmail REST API:

- Server: `https://gmail.googleapis.com`
- Port: `443`
- SSL: `On`
- Authorization URL: `https://accounts.google.com/o/oauth2/v2/auth`
- Token URL: `https://oauth2.googleapis.com/token`
- Scope: `https://mail.google.com/`
- OAuth Add Secret To Request: `On`
- Client Secret: **required**

> ***Important.*** *Google requires the Client Secret even for desktop apps. Do not leave the OAuth Secret field empty.*

4 Scenarios — Reference

For Gmail, only one scenario is supported:

4.1 Delegated + Authorization Code

Permission type	Delegated permissions
OAuth flow	Authorization Code
Web authentication	Yes, user logs in via browser
Refresh token	Yes, used for automatic refresh

The application acts on behalf of the signed-in user. The user completes web authorization once, after which Mailit uses the refresh token to automatically refresh access.

5 Account Types — Reference

Google Cloud Console defines two types of users:

- **External (Personal):** Any Google account (`@gmail.com`). Your app will be in "Testing" mode, limited to 100 test users, until you submit it for Google's verification.
- **Internal (Google Workspace):** Only users within your organization's domain. Does not require Google verification.

6 Application Registration — Reference

To use OAuth with Gmail, you must register a project in the Google Cloud Console. The process differs from Microsoft Entra.

1. Go to `https://console.cloud.google.com/`.
2. Create a new project.
3. Go to **APIs & Services** → **OAuth consent screen**. Choose **External** (for personal accounts) and fill in the required fields.
4. Go to **Credentials** → **Create Credentials** → **OAuth client ID**.
5. Select **Desktop app** as the application type.
6. Copy the **Client ID** and **Client Secret**.

> *You do not need to set a Redirect URI manually in the Google Cloud Console. "Desktop app" automatically allows `http://localhost`.*

7 Permissions — Reference

Google uses scopes to define access. Mailit uses a single broad scope for Gmail:

- `https://mail.google.com/` — Full access to the mailbox, including reading, sending, and deleting emails.

> *Because this is a broad scope, Google may display an "unverified app" warning when you authorize Mailit. This is normal for internal or testing apps.*

8 Mailit Connection Fields — Reference

Field	Description
OAuth Provider	Select the preset.
OAuth Client ID	Application (client) ID.
OAuth Secret	Client Secret Value. Required only for Application and Google templates.
User Name	Email of the mailbox. For shared mailbox — email of the shared mailbox.
Server	Filled in by the preset.
Server Port	Set automatically by the `Set Server Port` script.
SSL	Filled in by the preset.
OAuth Authorization URL	Filled in by the preset.
OAuth Token URL	Filled in by the preset.
OAuth Scope	Filled in by the preset.
OAuth Add Secret To Request	`Off` for Microsoft Delegated; `On` for Application and Google.
OAuth Refresh Token	Filled in automatically after authorization (Scenario 1 only).
OAuth Token Expires In	Filled in automatically after authorization (Scenario 1 only).

9 Server Addresses and Ports — Reference

9.1 Server Addresses

Transport	Server
Microsoft IMAP / POP / SMTP	`outlook.office365.com`
Microsoft Graph	`https://graph.microsoft.com/v1.0`
Google IMAP	`imap.gmail.com`
Google POP	`pop.gmail.com`
Google SMTP	`smtp.gmail.com`
Gmail API	`https://gmail.googleapis.com`

9.2 Ports

Protocol	SSL	Port
IMAP	`On`	993
IMAP	`Off`	143
POP	`On`	995
POP	`Off`	110
SMTP	`On`	465
SMTP	`Off`	587 (STARTTLS)

> ***These are recommendations, not a rigid standard.*** *If the server requires a different port, specify it manually in the* ***Server Port*** *field.*

9.3 Local Web Server and Mailit Assets

The Mailit plug-in includes a built-in simple static web server. It is used for two main purposes:

- **OAuth Authorization:** Capturing the redirect from the provider after a user signs in (using `index.html`).
- **HTML Editors:** Saving and loading HTML message bodies and images when using the Summernote and Automizy editors in the FileMaker Web Viewer.

> ***Protocol limitation.*** *The local web server only supports the* ***HTTP*** *protocol. It does not support HTTPS. All local addresses must start with* `http://localhost`*.*

9.3.1 Deploying Mailit Assets

To start the server, the **Mailit Assets** folder must be deployed next to the plug-in file.

- It is unpacked **automatically** when you install the plug-in via `Install Plug-in.fmp12`.
- If the folder is missing, you can deploy it separately by opening `Install Plug-in.fmp12` and clicking the **Install Assets** button.

Typical plug-in installation paths:

- **macOS:** `/Users/[USER_NAME]/Library/ApplicationSupport/FileMaker/FileMakerPro/[FM_VERSION]/Extensions`
- **Windows:** `C:\Users\[USER_NAME]\AppData\Local\FileMaker\FileMakerPro\[FM_VERSION]\Extensions`

9.3.2 Available URLs

By default, the server runs at `http://localhost:1234`. The following endpoints are available:

- `http://localhost:1234/` — Serves static files from the `../Mailit Assets/` directory. Only GET requests are supported.
- `http://localhost:1234/oauth/index.html` — Used for OAuth authentication redirects.
- `http://localhost:1234/editor/upload` — Used by HTML editors to upload images. Only POST requests are supported. Saves files to the `../Mailit Assets/editor/upload/` directory.
- `http://localhost:1234/editor/load` — Loads field content into the editor. Only POST requests are supported. Requires a `fieldName` parameter (e.g., `fieldName=Messages::Body`).
- `http://localhost:1234/editor/save` — Saves editor content back to FileMaker. Only POST requests are supported. Requires `scriptFileName`, `scriptName`, and `scriptParameter` (which must be encoded using `encodeURIComponent()`).

> *The local web server only supports the HTTP protocol and does not execute server-side scripts like PHP or Python. It is designed solely for static files and editor communication.*

9.3.3 Plugin Functions for the Local Server

The FileMaker scripts use the following plug-in functions to manage the local server:

- `Emai_LocalWebServerStart` — Starts the server. Parameters: `host` (default: "localhost"), `port` (default: "1234"), `path` (optional path to Mailit Assets).
- `Emai_LocalWebServerStop` — Stops the server.
- `Emai_LocalWebServerStatus` — Returns the current status and settings of the server.

If the local server cannot start, check that the plug-in has read/write permissions to the local drive, and that the port (1234) is not being used by another application.

10 Authorization Methods — Reference

10.1 Browser Login (Scenario 1)

Two options:

- **Open OAuth2 Window in FileMaker** — WebView inside FileMaker. **Get Code** button at the bottom.
- **Open OAuth2 in External browser** — default browser. No **Get Code** button. After successful authentication, you will see the message: *"Token received and inserted into the password field, you can now close the current browser tab."*

10.2 No Login (Scenario 2)

No browser login required. Mailit uses client credentials to obtain an access token directly.

11 Token Lifecycle and Automatic Refresh

11.1 Scenario 1 — Refresh Token

Microsoft returns **Refresh Token** and **Token Expires In** alongside the access token. Both can be viewed on the **OAuth 2.0 Additional Settings** tab.

*In the plug-in examples, the **Refresh Token** field is displayed in clear text for debugging. In production, store it in a safe place.*

Automatic refresh. The `Connect to the mail server` script checks `Accounts::OAuth Token Expires In`. If the token has expired, it sends a POST to `Accounts::OAuth Token URL` with `client_id`, `client_secret` (if enabled), `refresh_token`, and `grant_type=refresh_token`. The script updates `Accounts::Password`, `Accounts::OAuth Refresh Token`, and `Accounts::OAuth Token Expires In`, then retries the connection.

If refresh fails, re-authorize via the browser.

11.2 Scenario 2 — Client Credentials

No refresh token. The application requests a new access token each time, using `grant_type=client_credentials`.

12 Troubleshooting

12.1 Diagnostic Information

Collect the following before contacting support:

- Mailit build and FileMaker version.
- Operating system.
- The step where the error occurred.
- Exact error text.
- Contents of the **Last Session Transcript**.
- The plug-in log `Mailit_FM[FM_VERSION].log`.

Remove authorization codes, tokens, secrets, and message contents before sending diagnostics.

12.2 Local Web Server Errors (-30 / -61)

If the function `Emai_LocalWebServerStart` fails with error **-30** (incorrect option) or **-61** (file not found), it almost always means the local web server cannot start.

1. The **Mailit Assets** folder is missing next to the plug-in file.
2. FileMaker does not have read/write permissions to the **Mailit Assets** folder.
3. The port (default `1234`) is blocked by a firewall/antivirus or is already in use.

See section 9.3 for paths and how to fix this.

12.3 Common Gmail OAuth Errors

- **Google hasn't verified this app** — Because your app is in "Testing" mode, Google shows a warning. Click **Advanced** → **Go to Mailit (unsafe)**.
- `403: access_denied` — The Google account you are trying to sign in with is not listed in the **Test users** section of your OAuth consent screen. Add your email there.
- `redirect_uri_mismatch` — Make sure the Application Type in Google Cloud Console is set to **Desktop app**.
- `invalid_grant` — Refresh token revoked or expired. Re-authorization is required.

- `invalid_client` — Client ID or Client Secret is incorrect.
- `localhost` unreachable — The local web server failed to start. Check that the Mailit Assets folder is installed and that port 1234 is not being used by another application.