Modern-auth relay for legacy devices

Technical Specification

SMTP-365 documentation

Modern-auth relay for legacy devices

SMTP-365: Technical Specification (as built)

Version 0.2.0. This document describes the software as implemented. The original design brief is in `smtp-o365-relay-spec.md`; where the two differ, this document is authoritative.

0. Appliance additions in 0.2.0

Component	Module / file	Summary
Branding	`brand.py`	Product name, vendor, public signing key, trial length, setup-app client id
Licensing	`licensing.py`	`SMTP365-<b64 payload>.<b64 sig>` Ed25519 keys; payload has id, customer, email, issued, maintenance date, max_routes, features, product. `status()` combines the stored key with `first_boot` into licensed / trial / expired; jobs are held while not `delivery_allowed`
Update bundles	`updates.py`	tar.gz with `manifest.json` + `manifest.sig` + wheels + optional `post-update.sh`; signature and per-file SHA-256 verified before install; `min_maintenance` gate
Vendor tools	`vendor_cli.py` (`mfp-relay-vendor`)	keygen, issue, verify, build-update, verify-update
System agent	`agent.py` (`mfp-relay-agent`), `agentclient.py`	Root service on `/run/mfp-relay/agent.sock` (0660 root:mfp-relay), newline-delimited JSON commands with strict argument validation and no shell: system_info, get/apply_network (systemd-networkd), set_hostname, set_timezone, set_root_password, apply_firewall (nftables, validated before load), restart_services, tls_selfsigned/install/info, update_info/apply, backup_create/restore, reboot, shutdown, journal
Managed settings	`config.py` `MANAGED_KEYS`, `db.get_managed_config`	Selected `config.yaml` keys stored as `cfg:<dotted>` settings and overlaid at load (`load_effective_config`)
Microsoft 365 onboarding	`m365setup.py`	Device-code sign-in with the vendor's public setup app; creates/reuses the customer app, service principal, app-role assignments, secret, site permission; stores credentials; background thread polled by the browser
Web	`web/setup.py`, `web/system.py`, `web/m365.py`	Wizard sequencing (`WIZARD_STEPS`), System pages, Microsoft 365 page; `setup_complete` setting gates the wizard; password changes also set the root password via the agent
Appliance files	`appliance/`	`provision.sh`, nginx TLS proxy config, units (agent, console banner timer, first-boot), `preseed.cfg`, `build-wheelhouse.sh`, `build-iso.sh`, `build-images.sh`

Appliance layout: nginx on 443/80 proxies to waitress on 127.0.0.1:8080; SMTP binds 0.0.0.0 and relies on nftables plus `smtp.allowed_sources`; network is systemd-networkd (`/etc/systemd/network/10-smtp365.network`); TLS material in `/etc/mfp-relay/tls/`; uploads in `/var/lib/mfp-relay/uploads/`; backups in `/var/lib/mfp-relay/backups/`; first-boot marker `/var/lib/mfp-relay/.firstboot-done`.

1. Scope

mfp-relay receives email over SMTP from printers on a local network and delivers each message to Microsoft 365 through Microsoft Graph, either as an email sent from a tenant mailbox or as a PDF uploaded to a SharePoint folder. It includes an administrative web interface and command-line tools.

Out of scope: relaying to non-Microsoft destinations, delegated (user) Graph permissions, OCR, TLS or authentication on the inbound SMTP side, multiple tenants.

2. System architecture

2.1 Processes

Process	Entry point	Role
`mfp-relay-smtp`	`mfp_relay.cli:main_smtp`	SMTP listener (aiosmtpd) plus a single worker thread that delivers spooled jobs and sweeps for retries
`mfp-relay-web`	`mfp_relay.cli:main_web`	Flask administrative UI served by waitress (falls back to Flask's server if waitress is absent)
`mfp-relay-admin`	`mfp_relay.cli:main_admin`	One-shot maintenance commands

The two services are independent systemd units. They share state only through the SQLite database and the spool directory. Routing and settings are read from the database on every use, so changes made in the web UI apply to the next message without a restart.

2.2 Modules

Module	Responsibility
`config.py`	YAML configuration, environment overrides, loading of the systemd environment file
`db.py`	SQLite schema and access: settings, routes, admin account, job log; password hashing and policy
`parser.py`	RFC 5322/MIME parsing into `ParsedMessage`; attachment extraction; filename sanitising
`router.py`	Recipient classification into jobs
`pdf.py`	Image to PDF conversion
`graph.py`	MSAL token acquisition, Graph HTTP with error classification, sendMail, SharePoint upload, site/drive/folder browsing
`spool.py`	Durable on-disk job queues (pending, held, failed)
`jobs.py`	Job execution, retry/fail/hold outcomes, administrator alerts
`smtp_server.py`	aiosmtpd handler, worker thread, `RelayService` composition
`logging_setup.py`	JSON-lines file logging plus stderr for journald
`web/`	Flask application factory, blueprints, templates, folder-picker script

2.3 Runtime dependencies

aiosmtpd, msal, requests, img2pdf, Pillow, PyYAML, Flask, Flask-Login, bcrypt; waitress (optional, production WSGI server). Python 3.11 or later.

3. Message flow

```
SMTP client --MAIL FROM--> source IP checked against smtp.allowed_sources
```

```
            --RCPT TO ---> each recipient validated (see 3.1)
            --DATA ------> size check -> MIME parse -> route -> one spool entry per job
                           <-- 250 OK: queued <ids>
worker thread <-- queue -- job id
              read spool -> execute (send / upload) -> success: delete spool entry
                                                    -> transient error: reschedule
                                                    -> permanent error: move to failed/, alert
                                                    -> unmapped address: move to held/, alert
```

3.1 SMTP transaction rules

Stage	Condition	Response
MAIL FROM	peer IP not within `smtp.allowed_sources` (when the list is non-empty)	`550 5.7.1 Relaying denied from this address`
RCPT TO	address has no `@`	`553 5.1.3`
RCPT TO	address is `<local>@<sharepoint_domain>`, no route for `<local>`, and `routing.unmapped_action` is `reject`	`550 5.1.1 No SharePoint folder is mapped for …`
RCPT TO	ordinary address whose domain is outside `routing.allowed_recipient_domains` (when non-empty)	`550 5.7.1 Recipient domain not permitted by relay policy`
DATA	message larger than `smtp.max_message_bytes`	`552 5.3.4`, logged as a rejected job
DATA	message cannot be parsed or has no recipients	`554 5.6.0`, logged as a rejected job
DATA	no recipient produced a deliverable job	`550 5.7.1 No deliverable recipients`
DATA	otherwise	`250 OK: queued <short ids>`

aiosmtpd's own line-length and size limits also apply; the data size limit is set to `max_message_bytes` plus 1 MiB so that the relay's own 552 response, rather than aiosmtpd's, is what the printer normally sees.

Recipient addresses are lower-cased and de-duplicated. Envelope addresses take precedence over header addresses; headers are used only when the envelope is empty.

3.2 Routing

For each message, `router.route_recipients` produces a list of decisions:

- Every recipient at `routing.sharepoint_domain` (default `sharepoint.local`)

becomes its own job. If a route exists for the local part (case-insensitive) the action is `sharepoint_upload`; otherwise `unmapped`.

- Every other recipient is checked against `routing.allowed_recipient_domains`

(exact domain or subdomain match; empty list allows all). Disallowed recipients produce a `disallowed` decision, which is logged as rejected and not spooled.

- All remaining ordinary recipients are combined into one `send_mail` job so

a single Graph call sends to all of them.

3.3 Job execution

`jobs.JobProcessor.process(job)`:

1. Read the raw message from the spool and parse it.

2. If the action is `unmapped`, re-check for a route. If one now exists the

action becomes `sharepoint_upload`; otherwise the job is moved to `held/` and, on first hold, an alert is sent.

1. Execute:

2. `send_mail`: choose the sender (3.4), send via Graph with all

attachments as received.

- `sharepoint_upload`: look up the route again, convert attachments (4),

upload each file. Filenames already recorded in the job's `completed` list are skipped, so a retry after a partial failure does not create duplicates.

1. Outcome handling:

2. Success: job log updated, spool entry removed.

3. `GraphError` with `transient=True` and attempts remaining: reschedule

(3.5), status `retrying`.

- Any other `GraphError`, `PermanentJobError`, or unexpected exception:

move to `failed/`, status `failed`, alert sent.

3.4 Sender selection (send_mail)

In order:

1. If `sender_override_enabled` is `1` and `sender_override_upn` is set, that

mailbox.

1. Otherwise the message's envelope/header From address, if it contains `@`.

2. Otherwise `alert_sender_upn`.

3. Otherwise the job fails permanently.

The chosen mailbox must be permitted by the Exchange Application Access Policy, or Graph returns 403 and the job fails.

3.5 Retry policy

Transient conditions: HTTP 408, 429, 500, 502, 503, 504; network errors; MSAL `temporarily_unavailable` / `server_error`.

Delay before attempt *n* (1-based, after the *n*th failure): `min(base_delay_seconds × 2^(n−1), max_delay_seconds)`, raised to any `Retry-After` header value. With the defaults (120 s base, 1800 s cap, 5 attempts) the schedule is 2, 4, 8, 16 minutes, then failure after the fifth attempt, roughly one hour in total.

The worker sweeps the pending queue every `retry.poll_interval_seconds` (default 30) and processes every job whose `next_attempt_at` has passed.

3.6 Alerts

`JobProcessor.alert(subject, body)` sends an email from `alert_sender_upn` (falling back to `sender_override_upn`) to `admin_alert_email` via Graph. If either address is missing, or Graph itself fails, the alert is written to the log at ERROR level instead. Alerts are sent when a job is held for the first time and when a job fails permanently.

4. PDF conversion

Applied only to `sharepoint_upload` jobs. Email jobs forward attachments unchanged.

Attachment	Detection	Result
PDF	content type `application/pdf`, `.pdf` extension, or `%PDF-` magic	Passed through; `.pdf` extension ensured
Image	content type in the image set or extension `.tif .tiff .jpg .jpeg .png .bmp .gif`	All images in the message are merged, in order, into one PDF, one page per image or TIFF frame
Other	anything else	Passed through unchanged

Conversion uses img2pdf (lossless embedding). Images with alpha or palette modes are first flattened to RGB PNG via Pillow because img2pdf rejects them. If img2pdf fails for any reason, Pillow renders the PDF instead.

The merged PDF is named after the first image's base name, or after the message subject when the image had no real name (the parser assigns `scan-N` to unnamed parts). Whitespace becomes underscores and the name is sanitised (5.3).

5. Data model

5.1 SQLite database (`<data_dir>/mfp-relay.sqlite3`)

WAL journal mode, file mode 0600. When a root-run CLI creates the file or its `-wal`/`-shm` companions, ownership is transferred to the data directory's owner so the service user can open them.

`settings (key, value)` — string key/value store. Keys:

Key	Meaning
`tenant_id`, `client_id`, `client_secret`	Graph app-only credentials (overridable by environment; see 7.3)
`sharepoint_site_url`, `sharepoint_site_id`, `sharepoint_site_name`	The configured site; the id is resolved from the URL via Graph
`sender_override_enabled` (`0`/`1`), `sender_override_upn`	Fixed sender policy
`admin_alert_email`, `alert_sender_upn`	Alerting
`web_secret_key`	Flask session signing key, generated on first web start

`routes` — one row per `@sharepoint.local` address:

Column	Notes
`local_part`	unique, case-insensitive
`site_id`, `site_name`	site the folder lives in
`drive_id`, `drive_name`	document library
`folder_id`, `folder_path`	destination folder item id and display path
`description`	free text
`created_at`, `updated_at`	ISO 8601 UTC

`admin_user` — single row: `username`, bcrypt `password_hash`, `must_change_password`, `updated_at`.

`jobs` — audit log, pruned hourly to the newest 5000 rows: `id`, `created_at`, `updated_at`, `source_ip`, `mail_from`, `recipients`, `subject`, `attachment_count`, `total_bytes`, `action`, `status`, `attempts`, `graph_status`, `detail`. Statuses: `pending`, `retrying`, `success`, `failed`, `held`, `rejected`.

5.2 Spool (`<spool_dir>/{pending,held,failed}/`)

Each job is `<id>.json` (metadata) and `<id>.eml` (raw message). Files are written to a temporary name, fsynced and renamed, so a crash never leaves a partial job. A `.json` without its `.eml`, or an unparsable `.json`, is ignored. Metadata fields: `id`, `action`, `recipients`, `mail_from`, `source_ip`, `subject`, `created_at`, `attempts`, `next_attempt_at`, `last_error`, `completed` (uploaded filenames).

5.3 Filenames

Attachment names have path separators, `: * ? " < > |` and control characters replaced with `_`, leading/trailing dots and spaces stripped, are truncated to 200 characters, and default to `scan-N` when empty. An extension is added from the content type when missing. Uploads use Graph's `@microsoft.graph.conflictBehavior=rename`, so an existing name yields `name 1.pdf` rather than an overwrite.

6. Microsoft Graph integration

6.1 Authentication

MSAL `ConfidentialClientApplication` with client-credentials flow against `https://login.microsoftonline.com/<tenant>` for scope `https://graph.microsoft.com/.default`. Credential is a client secret or, when `graph.certificate.path` and `thumbprint` are set, a PEM private key. MSAL caches the token in memory and treats it as expired five minutes before its real expiry, so refresh happens transparently. The application object is rebuilt if tenant, client id or credential change.

6.2 Calls used

Purpose	Request
Send, all attachments ≤ 3 MB	`POST /users/{upn}/sendMail` with base64 `fileAttachment`s
Send, any attachment > 3 MB	`POST /users/{upn}/messages` (draft) → per attachment either `POST …/attachments` or `POST …/attachments/createUploadSession` + chunked `PUT` → `POST …/send`
Upload ≤ 4 MB	`PUT /drives/{drive}/items/{folder}:/{name}:/content`
Upload > 4 MB	`POST …:/createUploadSession` then `PUT` chunks of 1600 KiB with `Content-Range`
Site resolution	`GET /sites/{host}:/{path}` or `GET /sites/{host}`
Libraries	`GET /sites/{site}/drives`
Folder browsing	`GET /drives/{drive}/root/children` or `/items/{id}/children`, folders only, paginated via `@odata.nextLink`
Create folder	`POST /drives/{drive}/items/{parent}/children`
Connection test	token acquisition; roles read from the JWT; `GET /sites/{site}` if configured

Required application permissions: `Mail.Send` and `Sites.Selected` (or `Sites.ReadWrite.All`). The connection test warns if the token lacks them.

6.3 Error classification

Non-2xx responses raise `GraphError(status, transient, retry_after)`. Transient statuses are listed in 3.5; everything else is permanent. Missing credentials raise `GraphConfigError`, which is never retried.

7. Configuration

7.1 `config.yaml`

Location: `$MFP_RELAY_CONFIG`, else `/etc/mfp-relay/config.yaml`. A missing file yields defaults.

Key	Default	Meaning
`smtp.bind_host`	`0.0.0.0`	Listen address; bind to the LAN interface
`smtp.port`	`2525`	Listen port; 25 requires `CAP_NET_BIND_SERVICE` (granted by the unit)
`smtp.hostname`	`mfp-relay.local`	Name in the SMTP banner
`smtp.allowed_sources`	`[]`	CIDRs allowed to submit; empty allows any that reach the socket
`smtp.max_message_bytes`	`26214400`	Reject larger messages
`paths.data_dir`	`/var/lib/mfp-relay`	Database location
`paths.spool_dir`	`/var/spool/mfp-relay`	Job queues
`paths.log_file`	`/var/log/mfp-relay/relay.log`	JSON log
`retry.max_attempts`	`5`	Attempts before failing
`retry.base_delay_seconds`	`120`	First retry delay
`retry.max_delay_seconds`	`1800`	Delay cap
`retry.poll_interval_seconds`	`30`	Spool sweep interval
`routing.sharepoint_domain`	`sharepoint.local`	Pseudo-domain for uploads
`routing.unmapped_action`	`hold`	`hold` or `reject`
`routing.allowed_recipient_domains`	`[]`	Email recipient allow-list; empty allows all
`web.bind_host`	`127.0.0.1`	Web UI listen address
`web.port`	`8080`	Web UI port
`web.secret_key`	`""`	Flask session key; blank means generated and stored in the DB
`graph.authority_host`	`https://login.microsoftonline.com`	
`graph.endpoint`	`https://graph.microsoft.com/v1.0`	
`graph.timeout_seconds`	`60`	Per-request HTTP timeout
`graph.certificate.path` / `.thumbprint`	`""`	Certificate authentication
`log_level`	`INFO`	Root log level

7.2 Environment file

`$MFP_RELAY_ENV_FILE`, else `/etc/mfp-relay/mfp-relay.env`, is read by systemd for the services and by the code itself for CLI runs. `KEY=VALUE` lines; `#` comments; optional `export`; single or double quotes stripped. Existing environment variables are never overwritten.

7.3 Environment variables

Variable	Effect
`MFP_RELAY_CONFIG`, `MFP_RELAY_ENV_FILE`	File locations
`MFP_RELAY_TENANT_ID`, `MFP_RELAY_CLIENT_ID`, `MFP_RELAY_CLIENT_SECRET`	Override the database settings; the web UI shows them read-only
`MFP_RELAY_DATA_DIR`, `MFP_RELAY_SPOOL_DIR`, `MFP_RELAY_LOG_FILE`	Path overrides
`MFP_RELAY_SMTP_PORT`, `MFP_RELAY_WEB_PORT`, `MFP_RELAY_WEB_SECRET_KEY`	Port and key overrides

8. Web interface

8.1 Routes

Method, path	Purpose
`GET/POST /login`	Sign in; 10 failures per source IP in 5 minutes returns 429
`POST /logout`	Sign out
`GET/POST /password`	Change password; enforced for every other page while `must_change_password` is set
`GET /routes`	Route list
`GET/POST /routes/new`, `/routes/<id>/edit`	Route form with folder picker
`POST /routes/<id>/delete`	Delete
`GET/POST /settings`	Credentials, site, sender policy, alerts
`POST /settings/test`	Connection test
`GET /status`	Counters, token state, held/failed/pending jobs, recent job log
`POST /jobs/<id>/requeue`, `/jobs/<id>/discard`	Spool actions
`GET /api/drives`	Libraries of the configured site (JSON)
`GET /api/browse?drive_id=&item_id=`	Child folders (JSON)
`POST /api/folders`	Create folder (JSON)

8.2 Security controls

- Session cookie is HttpOnly, SameSite=Lax, signed with the stored key.
- Every state-changing request must carry the session's CSRF token as a form field or `X-CSRF-Token` header.
- Password policy: at least 10 characters, letters and digits, not in a blocklist of common passwords (including the default `changeme`), not containing the username.
- The client secret is write-only: it is never rendered back to the browser

and a blank field on save keeps the existing value.

- Request bodies are capped at 1 MiB.
- No HTTPS: the interface is intended for LAN use behind the firewall rules.

9. Logging

`paths.log_file` receives one JSON object per line, reopened automatically after logrotate (WatchedFileHandler). Stderr receives a plain-text copy for journald. Fields: `ts`, `level`, `logger`, `msg`, plus any of `job_id`, `source_ip`, `mail_from`, `recipients`, `subject`, `attachment_count`, `total_bytes`, `action`, `graph_status`, `status`, `attempt`, `detail`, `exc`. Libraries msal, urllib3, werkzeug and aiosmtpd's mail.log are limited to WARNING.

10. Command-line interface

All commands accept `-c/--config PATH` and `--version`.

Command	Purpose
`mfp-relay-smtp`	Run the SMTP service (foreground)
`mfp-relay-web [--debug]`	Run the web UI (foreground)
`mfp-relay-admin show-config`	Effective YAML config, DB settings (secret masked) and routes as JSON
`mfp-relay-admin reset-password [--password P] [--final]`	Set the admin password; without `--final` a change is forced at next login
`mfp-relay-admin test-graph`	Acquire a token, list granted roles, read the configured site; exit 1 if roles are missing
`mfp-relay-admin settings show`	Print settings (secrets masked)
`mfp-relay-admin settings set key=value …`	Set any settings key except the web key
`mfp-relay-admin settings resolve-site URL`	Set the site URL, id and name via Graph
`mfp-relay-admin send-test TO [--from F] [--file PATH] [--host H] [--port P]`	Submit a test message through the local SMTP listener
`mfp-relay-admin run-due`	Process due spool jobs once

11. Deployment artefacts (`deploy/`)

File	Purpose
`install.sh`	Packages, service user, directories, venv in `/opt/mfp-relay`, units, logrotate
`configure-server.sh`	Runs `install.sh`, writes `config.yaml` for a given IP and subnets, installs nftables rules, starts services; values overridable via environment variables
`Register-MfpRelayApp.ps1`	Entra app registration, consent, credential, site grant, Exchange access policy; PowerShell 5.1 and 7
`mfp-relay.service`, `mfp-relay-web.service`	systemd units: unprivileged user, `ProtectSystem=strict`, `Restart=always`, unlimited restarts for late DHCP leases
`mfp-relay.env.example`	Environment file template
`logrotate-mfp-relay`	Daily rotation, 30 kept

File	Purpose
`nftables-example.conf`	Firewall template

12. Limits and known constraints

- One worker thread: jobs are delivered serially. Adequate for printer volumes; not intended for bulk mail.
- Messages and attachments are held in memory during processing; the practical ceiling is governed by `smtp.max_message_bytes` (25 MB default).
- Inbound SMTP has no TLS or authentication by design; the controls are the firewall and `smtp.allowed_sources`.
- Single administrator account.
- One SharePoint site per installation (routes may target any library or folder within it).
- The email path forwards attachments unconverted; PDF conversion applies to SharePoint uploads only.
- Clients must send CRLF line endings per RFC 5321; aiosmtpd rejects overlong lines with 500.

13. Testing

`pytest` runs 61 tests without network access: parser, router, PDF conversion, spool durability, database and password policy, job outcomes (success, retry, exhaustion, permanent failure, hold and re-queue, partial upload progress), the web UI (forced password change, CSRF, routes CRUD, settings, connection test, folder picker API, status actions), configuration loading, and end-to-end SMTP submission over a real socket against a fake Graph client.