
Filum, technical: what IT wants to know before rollout

Permissions, data paths, archive format and verifiability. A sheet for the IT of an architecture or planning office.

As at 26.09.2026. Every statement here comes from the code as it stands in the tree today, not from a brochure. bimover support provides the setup sheets this paper refers to on request. Not legal advice.

Filum is not published yet. The macOS version is in preparation for the App Store, the Windows version is on hold. This sheet describes the state of the program, not a delivery date.

1. What Filum does

An architecture or planning office runs its project correspondence in Microsoft 365. Each person files their project mail into an Outlook folder or has it assigned by rules they set themselves (contact, project number and name, evaluated on their Mac). Filum reads that folder, or with rules the inbox and sent items, through Microsoft Graph and puts what belongs to the project into a folder on the office NAS: the originals as `.eml`, the attachments unpacked, the conversations as HTML, plus an overview with search that works straight off the disk. A run captures every filing and every deletion in a record covering the entire holding, chained to the previous one and timestamped by a third party.

After the first run, Filum asks once whether it may carry on in the background. For whoever agrees, Filum then archives their projects every two hours on its own (section 9).

Three properties determine everything else:

- Nothing is kept at bimover. There is no bimover server holding mail and no database holding an archive. The archive is a folder on a medium the office chooses itself.
- Filum works with the sign-in of the person using it on their computer, in the background too. There is no service account and no tenant-wide read access.
- The archive is readable without Filum and verifiable without Filum. Neither is a statement of intent; both sit as files in the folder.

2. Permissions

Delegated, not application-level

Filum signs in through the multi-tenant registration «Filum Mailspace», a public client with no secret on the workstation. On Windows a device code runs, on macOS a sign-in window. The app then sees exactly the mailboxes and folders the signed-in person may open anyway, including shared and delegated ones.

The alternative would be application-level access. It would need `Mail.Read` for the whole tenant, that is read access to every mailbox in the office, and a secret stored somewhere. Filum has neither, and that is why rollout needs no project.

Permission	What for
<code>Mail.ReadWrite</code>	read the project folder and fetch raw bytes; for a project with rules, read the inbox and sent items with their text to assign messages to projects; while collecting, show the newest messages from the inbox and sent items and move accepted ones into the project folder; in the background, read the inbox headers (section 3)
<code>Mail.ReadWrite.Shared</code>	the same for shared and delegated mailboxes
<code>User.Read</code>	display name and address of the signed-in person, and from it the own domain
<code>openid</code> , <code>profile</code> , <code>email</code>	who is signed in; Filum uses it to connect the bimover account without a second sign-in
<code>offline_access</code>	keep the sign-in beyond a program restart

Write permission is on the list because collecting moves messages. Filum deletes nothing in any mailbox, and it writes no message.

When the tenant blocks third-party apps

If an organisation blocks user consent, the first sign-in fails. Then, and only then, Filum shows the prompt with the address for the one-time approval by whoever administers the accounts. Afterwards the app stands as an enterprise application in the organisation's own tenant and applies to everyone in the office.

There is deliberately no separate registration per customer. It would not narrow the access; it would only push a form onto the customer.

3. Data paths

Filum talks to four counterparts on the network and, besides them, asks the computer's name service for public keys. It talks to nothing else.

Destination	What goes there	When
<code>login.microsoftonline.com</code>	sign-in, token renewal	when signing in and afterwards in the background
<code>graph.microsoft.com</code>	folder and message queries carrying the access token; before each run one request without a token that only shows whether Microsoft is reachable and what time it reports	during a run, while collecting, in the background
<code>www.bimover.ch</code>	sign-in to the bimover account, licence evidence, request for free licences, report after the run, support	table below
Timestamp service: default OpenTimestamps (<code>a.pool.opentimestamps.org</code> , <code>b.pool.opentimestamps.org</code> , <code>a.pool.eternitywall.com</code> , <code>ots.btc.cataallaxy.com</code>), optionally <code>tsa.swisssign.net</code>	the hash of one record, nothing else	on every run that files something or records a deletion, otherwise at the first run of a day
The computer's name service, and through it the name server of the sender's domain	selector and domain of a signature, no address and no content	during a run and in the background

What goes to bimover

When	What goes there
When the bimover account is connected	Microsoft's sign-in token with the person's name, address and identifier and the identifier of their organisation; a device identifier with a public device key, the computer's name, platform and program version
At start-up, when reloading and on «Check»	the sign-in and the app identifier; the signed licence evidence and the account status come back
While setting up and on the project page	one SHA-256 instead of the address for each mailbox the project archives from; nothing is reserved
Before every run, in the background too	the same hashes; now the licences are reserved
After a run	the same hashes, an archive identifier, identifier and time of the run, the checksum of the latest record, how many messages the archive holds and how many records its chain counts
Only when someone writes to support	subject and text of the request, name and address for the reply, platform, version, language and a resettable installation identifier; with explicit release, the diagnostic report

The hash of a mailbox is pseudonymous, not anonymous. The salt is the same for everyone, so that the same mailbox in two projects stays one licence: whoever knows the address can recompute the hash, but the address cannot be read off the hash alone. The archive identifier is created when the archive is set up and carries neither a project name nor an address. Subject lines, correspondents' addresses, file names and contents are not provided for in these reports, and a fixed list of forbidden field names in the tests holds that in place. The sample project reports nothing.

The report after a run has a second purpose that may interest IT: everything proving an archive is unchanged otherwise sits in the same folder that whoever wanted to change it can write to. The server remembers, per archive, the highest number of records and messages it has seen. If the same archive later reports fewer, Filum says so.

The diagnostic report leaves only when someone releases it to support, and the app shows the whole text first. It names no subject line and no correspondent's address. It does name the user's own account address, so support knows whom to answer, plus project name and number, only the last folder of the storage path, and how long the last steps took. The privacy policy for Filum is at www.bimover.ch/filum/privacy.

Requests to Microsoft go exclusively to graph.microsoft.com. Even the continuation address in a response, which Filum uses to page through results, is checked against that first; if it points elsewhere, Filum does not follow it and says so in the run report.

Public keys and the inbox

Many business emails carry a signature of their sender's domain (DKIM). It can only be checked with the public key the domain publishes in DNS, and domains change their keys. Filum therefore looks the key up through the computer's name service when filing and puts it into the archive. The lookup names a selector and a domain, such as `selector1._domainkey.example.ch`. It is visible to the office's name service or its provider and to the name server of the sender's domain. There is deliberately no third party answering DNS over HTTPS.

When Filum runs in the background, after each run it also fetches the headers of the newest 200 messages in the inbox of every mailbox a project archives from. This request fetches no text and no attachments. The headers also contain sender and subject; Filum evaluates only the `DKIM-Signature` line and keeps nothing of the rest. The keys go into a store on the computer, inside the app's container. Only what an archived message names moves into the archive; what nobody needs for twelve months is dropped. So the archive holds no domain for which nothing was filed.

While collecting, that is only when someone opens it in the app, Filum reads subject, sender and date of the newest messages in the inbox and in sent items, to suggest what belongs to the project. That stays on the computer. The sample project asks neither Microsoft nor the name service.

4. Where the archive sits

A separate share on the NAS, for example `Mailarchiv`. Projects are subfolders inside it. There are no permissions per project, and that is a decision rather than an omission: an office of six does not maintain a permission matrix across forty projects, and what nobody maintains is wrong after the second staff change.

The four rules of the setup:

1. A separate share for the archive, not inside the project storage. It can then be backed up differently from the work in progress, and restoring something in it touches no drawings.
2. Everyone writes with their own account; the staff group gets read and write. No shared archive account: Windows does not allow two simultaneous connections to the same server with different accounts.
3. Snapshots run automatically and belong to the NAS administration. For the staff they are invisible, and nobody on the staff administers the NAS.
4. Deleting happens only through the app, because the delete function leaves a record instead of a gap.

This holds because the check names every missing or altered file and the snapshot brings it back. That is exactly how the law treats changeable media. An earlier model with letterbox permissions (create yes, delete no) was abandoned on 14.09.2026: laborious to set up, on some devices enforceable only over SMB, and whoever had created a file could delete it anyway.

Plan for snapshot space: originals and attachments are only ever added and cost nothing extra in a snapshot. A run that finds nothing new writes only a small run header, and at the first run of a day a sign

of life is added: record, log and timestamp, a few kilobytes together. A run that adds something, on the other hand, rewrites the archive state, the overview, the search index of the reading view and `verzeichnis.csv` in full, and these grow with the number of messages. A project in which nothing happens any more therefore takes up hardly any new space in the snapshots, even though the background looks at it every two hours. How much a run writes at 2,000 and at 20,000 messages, and what that means for a retention plan, is in the setup sheet.

During a run a lock file sits in the archive so that two runs do not write at once. If a run breaks off hard, the lock counts as orphaned after two hours and the next run takes over.

The steps for Synology and QNAP, including access logging, are in the setup sheet.

5. What sits in the folder

2019-0234_neubau-sonnenhof/	
Mails ansehen.html	overview with search and filters, the main entrance
Mails/	one archive file per message (PDF/A-2u), chronological
.filum/	hidden, everything else:
originale/	the originals as .eml, untouched, flat
verlaeufer/	one readable HTML per conversation
anhaenge/	unpacked, deduplicated by content
dkim/	public keys of the sender domains
verzeichnis.csv	one row per message, for a spreadsheet or text editor
liesmich.txt	what the archive is and how to search it
belege/	holding records and timestamps (.ots, .tsr) by year, anchors
pruefen.py	verifies the archive without Filum
verfahren.txt	procedural documentation
protokolle/	run logs, by year
pruefungen/	check logs

The distinction everything hangs on: the evidence is the `.eml` alone. It is the most original form available of a message, with the attachments inside it. Everything else is a reading copy and can be regenerated from the `.eml` at any time: the archive file in `Mails/`, the unpacked attachments, the reading view, the index. A deviation in an original is a hard finding, a regenerated archive file is not.

File names look like this:

```
2019-04-02 0914 Eingang - Anna Muster - Baugesuch Sonnenhof a5d21a19.eml
```

Date and time first, so that any file manager sorts chronologically; local time, because people search for the time that stood in Outlook; plain ASCII, because macOS decomposes file names and Windows composes them and the behaviour over SMB with mixed forms is undefined. The real subject with umlauts sits in the HTML and in the CSV. At the end of the name stands a short identifier derived from the message identity.

As an option, Filum files new messages without the payload of their attachments. The original is then re-assembled byte for byte from the skeleton and the parts under `.filum/anhaenge`, and check, extrac-

tion and the verification script all compute over the reassembled original. For an office where the same set of drawings sits in twelve mailboxes, that is the difference between twelve copies and one. A separate run converts already filed messages in either direction, without identifiers or records changing.

6. Evidence and verification

Every message has two identifiers. Its identity is a hash over the canonicalised form, not over the bytes: Exchange keeps MAPI and reassembles the MIME on every fetch, so a byte hash would invent new messages at every quarterly run. Alongside it sits the byte hash of the stored file, and that is what the check recomputes.

A run that files something writes a holding record: a hash over the entire holding, both identifiers per message, sorted, chained to the previous record. After a deletion, the next run writes one, with or without new mail. If nothing has changed, a run writes none; at the first run of a day, one is written all the same, as a sign of life, so that a timestamp attests that the archive lay there in this form on that day. On every record Filum obtains a timestamp and stores it raw next to it. The default is OpenTimestamps: four public calendars take the hash and anchor it in the Bitcoin blockchain within hours, with no contract and no provider anyone has to trust. SwissSign, a timestamp under RFC 3161, can be chosen instead. An OpenTimestamps receipt first names the calendars that still owe the block; today Filum does not fetch the finished confirmation itself, and the verification script names the command with which the open tool `ots` checks it on a copy. With a two-hour cycle, the background adds at most twelve records and stamps a day per project and computer, and one in a quiet project. If the timestamp service is down, the run finishes and a later one stamps afterwards.

Every record also names the public keys its run filed, with their hash. The timestamp thereby attests that this key lay there in this form on that day. Filum does not recompute the signature of a message itself; that is left to an expert witness with their own tools, and the key is ready for them.

Verification happens two ways that compute the same thing:

- in the app, on the project page,
- without Filum via `python3 -I .filum/pruefen.py`, which needs only the standard library and also recomputes the timestamps.

A parity test keeps both verifiers tied to each other so they cannot drift apart. The check has three outcomes rather than two: present, removed with a deletion record, missing without one. Without that distinction a lawful deletion would look like tampering.

7. Deleting

Deletion removes the original and the reading copy together with the attachments no other message needs, redacts the line in the record file and places a note with both identifiers next to it. The message is gone, the evidence that one was there remains, and the chain stays valid. Afterwards the sender and subject of the message no longer appear in any file name or in any file that Filum keeps about the mes-

sage. Only an older log that named the message because of a finding keeps that line: logs are evidence and stay as they were written. Hashes without an original are no longer personal data.

Two properties that matter in operation: a deletion that breaks off mid-write can be selected again in «Auskunft und Löschung» and finished with the timestamp from back then. And if a symbolic link sits on the path to a file the deletion would touch, no plan is created at all, and Filum says so. A template for the office's deletion concept is openly available at github.com/bimover/mailarchiv-vorlagen.

8. The Filum Server, if staff should read in a browser

Anyone who does not want to mount a drive sets up the Filum Server on the NAS: a small service that serves exactly the reading view already present in the archive. It cannot write, there is no endpoint for it, and it does not record who reads which message when. Access is one shared reading password for the whole office; whoever wants finer separation separates at the network.

Because it sees all archives, it can do what a single folder cannot. It finds every archive by its identifier, even deep inside a project folder, and the address in the browser names that identifier instead of a path. It searches the messages of all projects, by the same rule as the app. And once a day it checks every archive the way the app does under «Check», with the same verdict as `pruefen.py`; it shows the state for each project and writes nothing into the archive while doing so.

The key assurance stays: if the service fails or is never set up, every archive in the folder stays fully readable. The server is convenience, never a prerequisite. bimover support provides the Filum Server and its setup sheet on request.

9. Operation

Requirements. macOS 14 or newer. The Windows version requires Windows 10 version 1809, but it is on hold and not available today. Plus a Microsoft 365 account with an Exchange Online mailbox, an internet connection and write access to the archive share.

One run. The app lists the chosen folders, fetches the headers page by page, then the raw bytes in batches of twenty, builds the conversations, writes files, records and the reading view. Several mailboxes run alongside each other; within one mailbox at most three requests at a time, because that is where Outlook draws the line. If Microsoft throttles, Filum waits the stated time and continues. A second run recognises what is already there and fetches only what is new.

In the background. After the first run, Filum asks once whether it may carry on in the background. For whoever agrees, Filum registers with macOS as a login item. That is the app itself, without a window and without a Dock icon, not a second program. macOS wakes it every 30 minutes, and it starts a run every two hours, for all projects this person has set up; the sample project is not among them. Each computer delays its first run by up to two hours, so that not every computer in an office reaches for the NAS at once. The same lock and the same rules apply as for a run by hand: no network, licence or sign-in, no run. There is a notification only when someone has to resolve something, and then once, for instance when

the sign-in or the licence is missing. A run that stands for more than four hours ends itself, and the next one picks up the rest. Whoever declines is not asked again and can switch it on later in the settings. It can be switched off in Filum's settings and in System Settings under login items. It runs only while the person is logged in to the Mac.

After a break-off. An aborted run does not leave half an archive behind. Writing happens only once a file is complete, the archive state after the messages and records, and a small run header last of all. A new run picks up the work.

Annual review. Once a year the app shows, across all projects, which retention periods have expired, and records the review as text. Nothing is ever deleted automatically.

Network and account. Without a valid licence no new run starts, and without a connection none either. Existing archives are untouched by this: opening, reading, checking, extracting, disclosure and deletion always work, even without an account and without a network. If bimover.ch does not answer for a while, stored licence evidence carries until its grace period, and the app says how many days that still is.

10. Licensing

What is counted are archived mailboxes, not people who start a run. Anyone who has access to a shared mailbox and archives it needs a licence for it.

The calculation is tiered like income tax: the rate of a tier applies only to the mailboxes within that tier. That way the total cannot fall when a mailbox is added. As at 24.09.2026; the source is bimover's price table:

Mailboxes	per mailbox and year
1 to 10	CHF 50
11 to 20	CHF 40
21 to 30	CHF 30
from 31	CHF 25

The smallest invoice is CHF 250. An office with twenty mailboxes pays CHF 900 a year. The bimover AI answers questions in the Support app, and that is included in the licence; a person's time is charged as it is spent, in quarter-hour units. Purchasing happens on bimover.ch, not in the app.

11. Limits, stated openly

- Teams chats are not covered.
- Cloud attachments, that is OneDrive links instead of real file attachments, are archived as links. Graph provides no bytes for them.

- Drafts are skipped. They have neither a Message-ID nor transport headers.
- Filum does not prevent changes to the archive. It makes them detectable. Protection against accidental deletion comes from the NAS snapshot.
- Filum does not recompute the signature of a message itself. It files the key so that someone else can.
- Filum works in the background only on the Mac, and only while the person is logged in to the Mac.
- A conversion run across very many messages can take longer than the two hours after which a lock counts as orphaned. If another run then takes over, a guard stops the conversion run before it writes, and it has to be started again.
- Anyone routing the traffic to Microsoft through their own proxy with their own certificate authority can forge the time against which Filum measures the grace period. That is deliberate effort, not a wrong clock.

12. Where it is written more precisely

Question	Where
Archive format, identifiers, chain, limits	<code>.filum/verfahren.txt</code> and <code>.filum/liesmich.txt</code> , in every archive
Privacy at Filum	<code>www.bimover.ch/filum/privacy</code>
Regulations, register, deletion concept	as open templates at <code>github.com/bimover/mailarchiv-vorlagen</code>
Archive share, snapshots, restoring, access logging	setup sheet, from support on request
Filum Server on the NAS	Filum Server setup sheet, from support on request
Tamper protection in detail	from support on request