

Automated File Routing

User Manual

Deliver files. Save feeds.
Run on your schedule.

Local and network folders - remote servers
Outlook attachments - OneDrive folders

Windows desktop application | Version 1.0.7

Reference edition | October 20, 2026

NM Softworks

Contents

Important: unattended runs require Windows to stay signed in 4

Microsoft Store purchase license 4

First launch: agree to the software terms 4

Open Help in the application 5

Choose a source and delivery rule 5

Quick start: run and schedule a OneDrive feed 5

Upload files to OneDrive destinations 6

Understand OneDrive timestamps and results 7

Find text in Help 8

Main window menus 8

Saved Locations window 8

Connect a OneDrive account 8

Connect an Outlook online mailbox 10

What the application does 21

Inspect SFTP source files 22

Inspect HTTPS source files 22

Deliver from an HTTPS source 23

Deliver from an SFTP source 24

Inspect FTP and FTPS source files 26

Deliver from an FTP or FTPS source 27

Start the application 28

Keep running in the system tray 28

Run schedules with the application closed 29

First local-folder delivery	31
Send the same files to several destinations	31
Choose the right source and destinations	32
Replace existing folder files	32
Replace existing SFTP files	33
Archive delivered source files to Used Files	34
Named feeds (manual runs)	36
Saved Locations: save, edit, test and delete	37
Save a folder location	38
Save an SFTP location	38
Save an FTP or FTPS location	39
Save an HTTPS upload location	41
Schedule a named feed	43
Preview, Run Now and results	47
Delivery history and Clear History	48
Verify a new destination: upload, then skip	49
Restart verification	49
Troubleshooting	50
Stored data	51
Verification coverage	51
OneDrive: copy to an existing local folder	52
OneDrive sources in saved feeds	52
OneDrive to HTTPS and scheduled runs	54
OneDrive to FTP/FTPS, schedules and Background Runs	55

Updated October 20, 2026. Reference edition for version 1.0.7.

Important: unattended runs require Windows to stay signed in

DO NOT SIGN OUT OF YOUR WINDOWS USER ACCOUNT if you want unattended deliveries to continue. Keep the computer powered on and awake. **Lock the screen with Windows + L instead of signing out.** Signing out, sleeping or shutting down stops scheduled delivery, even when Background Runs is ON.

With **Background Runs enabled**, you may close the main application and leave the screen locked. With Background Runs OFF, keep the app open or hidden in the tray. The current application does not provide a Windows service that runs while nobody is signed in. After a restart, sign into the configured Windows account before expecting background deliveries.

Automated File Routing delivers files from local or network folders, remote servers, Outlook email attachments and OneDrive folders to your selected folders or remote destinations. Save recurring delivery setups as named feeds, preview what will happen, and run them manually or on a schedule, including while the application is closed when Background Runs is enabled. Delivery history records what was copied, replaced, skipped or failed so you can review each run.

Microsoft Store purchase license

The Microsoft Store edition is a paid app with no free trial. Use **Help → Store License** to see your purchase-license status, open Microsoft Store, or refresh after purchase. The Store listing shows the applicable price.

A purchased, active Store license is required for Preview, Run Now and scheduled delivery, including Background Runs. If purchase entitlement cannot be verified, new operations pause. A delivery already running may finish; files and saved feeds are not deleted. Help, history and configuration remain accessible if Windows permits the app to open. Refresh after purchasing, then review the next scheduled run. Background Runs resumes only with your existing app opt-in and Windows startup permission; refreshing never grants new startup consent. The computer must remain signed in and awake.

Windows supplies cached offline license information when available. If license access fails, open Microsoft Store under the acquiring account and refresh. The unpackaged Visual Studio development build does not use Store licensing. Install or update through the official [Microsoft Store listing](#). During a limited rollout, the listing may be restricted to designated testers. A locally signed test package is not the Store edition.

First launch: agree to the software terms

Before the main application opens, review the NM Softworks Software License Agreement and Terms of Use summary and its full-terms/privacy links. Check the acknowledgment box, then select **AGREE & CONTINUE**. The box starts unchecked and the button stays disabled until you check it. Closing the agreement window exits without opening the application.

Acceptance is recorded separately for Automated File Routing and for each Windows user, using Windows encryption. The local record includes the application version/build, agreement version/effective date/identifier, UTC acceptance time and a unique identifier. A valid current record avoids repeating the prompt on ordinary restarts; a missing, invalid or changed agreement record requires acceptance again. If saving fails, the application stays on the agreement screen. This is local evidence, not a permanent server-side log.

Background Runs cannot accept on your behalf and performs no scheduled delivery without a valid current acceptance record. After an agreement update, open the application and accept before relying on closed-app schedules. Multi-Pad

acceptance does not authorize this application. Existing feeds, settings, credentials and transferred files are not erased by this screen.

Open Help in the application

Choose **Help** → **User Guide** on the main window or press **F1** there to open this guide. Select a topic on the left to jump to its instructions, or use Find to search the full guide. The guide text wraps and can be selected and copied with Ctrl+C. Resize the Help window for more reading space. Close it with **Close** or the window X; the main application stays open.

Help uses the guide bundled with the application and works offline. Rebuilding includes the latest guide changes. Help is a separate reading window: opening it does not pause schedules, change selections, save a feed, or transfer files. Other configuration dialogs retain their existing scheduling behavior.

Choose a source and delivery rule

Choose the source that matches the files you want to deliver. A saved feed keeps its source and destinations; schedules use those saved choices.

Source	Supported destinations	Existing destination files	Source handling
Local/network or saved remote folder	Existing local/network folders, saved SFTP, FTP, FTPS or compatible HTTPS locations, and OneDrive upload destinations	Folder/server replacement is optional; OneDrive uploads always skip existing names in this first stage	Retain, or archive to Used Files when supported and selected
Outlook personal/work mailbox	Existing local/network folders and saved SFTP, FTP, FTPS or compatible HTTPS locations	Delivery records skip already delivered attachments; distinct attachments use available names or stable suffixes	Emails and attachments remain in the mailbox
OneDrive personal/work folder	Existing local folders and saved SFTP, FTP, FTPS or compatible HTTPS locations	Copy missing names; replace older same-name regular files; skip equal/newer destinations	OneDrive originals remain unchanged; no archiving or delivery receipts

OneDrive destination uploads are available for local/network and supported remote folder sources, using the separate rules below. Outgoing email is on hold and SharePoint libraries are excluded. OneDrive source workflows and the agreed first-stage OneDrive destination tests have passed user testing in the reported environments. Destination testing includes small/large uploads, same-name skips, in-app/tray scheduling, fully closed-app Background Runs, a remote-source route and controlled failure/reconnection. These reports do not establish every account type, remote protocol or third-party server configuration. Server capabilities still determine whether a particular remote location can support the required checks.

Quick start: run and schedule a OneDrive feed

1. Open **Setup** → **OneDrive Source Accounts....** Add or select a personal/work account, choose its existing folder, then return to the main window and use **Use Saved....** OneDrive and Outlook keep separate connections.
2. Add existing local folders with **Add Folder....**, or use **Add Saved Location** for SFTP, FTP, FTPS or HTTPS destinations. Keep archiving and generic replacement choices off; OneDrive uses its own newer-file rule.

3. Click **Preview**. **Would Copy** means the filename is absent; **Would Replace** means OneDrive is newer; **Skipped/collision** means the destination is equal/newer or otherwise protected. A failed row needs review. Preview reads metadata and does not transfer files or prove write permission.
4. Click **Run Now**, then review **Feeds** → **Delivery History** → **View Delivery Results** and destination contents. Direct files only are delivered; subfolders are excluded. OneDrive originals remain unchanged. Limits are 1 GiB per file and two minutes per feed run.
5. Use **Feeds** → **Save Feed**, then **Feeds** → **Saved Feeds** → **Schedule....** Choose Interval, Daily, Weekly or One Time and enable automatic runs. Microsoft-source connection retries are fixed at zero. Close configuration windows so in-app scheduling can run.
6. Keep the app open or hidden in the tray for in-app runs. For runs with the application fully closed, also enable/update **Setup** → **Background Runs**. Remain signed into Windows and keep the computer awake. Scheduled OneDrive sign-in must work silently; reconnect manually if required.
7. Resaving an existing feed disables its schedule. Review and re-enable it intentionally. Failed/interrupted runs disable the schedule for review. Inspect History and destination/recovery files before retrying an uncertain result.

Upload files to OneDrive destinations

Both OneDrive account windows are in **Setup: OneDrive Source Accounts** reads files FROM OneDrive, and **OneDrive Destination Accounts** uploads files TO OneDrive. For everyday selection, **Use Saved** under Source offers each source account's last chosen folder, alongside saved local/network and remote sources. Accounts without a chosen folder are omitted; choose one in OneDrive Source Accounts first. **Add Saved Location** under Destination offers previously saved OneDrive upload folders alongside other saved destinations. Source and destination connections remain separate even when they use the same Microsoft account.

In either account window, **Choose OneDrive Folder... → Use This Folder** saves the folder for later selection. Source folders become available through **Use Saved**; destination folders become available through **Add Saved Location**, without needing to add them to the current feed. **Use As Source** and **Use As Destination** are optional shortcuts that also select the folder for the current feed. Both Setup-menu account windows offer the matching shortcut. Signing in alone does not choose a folder.

Disconnecting a OneDrive destination account removes its sign-in connection, while saved folder entries and existing feed references remain. **Add Saved Location** hides those entries until the matching destination account is connected again. To remove a saved folder entry itself, use **Setup** → **Saved Locations** → **Delete Location**. **Remove Selected** and **Clear Selections** on the main window only remove current selections. None of these actions deletes cloud files.

Select **Setup** → **OneDrive Destination Accounts....** This opens **OneDrive Destination Accounts**, separate from read-only OneDrive source accounts. Add a Personal or Work / School account with **Sign In And Save**, then choose and save an existing OneDrive folder. It becomes available through **Add Saved Location**; **Use As Destination** also selects it for the current feed. Microsoft requests delegated **Files.ReadWrite** permission. This permission covers the signed-in user's files more broadly than the selected folder; the application pins each destination to your chosen account, drive and folder. Company administrator approval may be required. SharePoint libraries and shared shortcuts are excluded.

This first upload stage supports local/network folder sources and existing saved SFTP/FTP/FTPS/HTTPS folder sources. Outlook mailbox and OneDrive sources cannot yet target OneDrive. Nonempty files up to **1 GiB each** are supported; empty files report Failed. Only direct source files are routed. Preview checks destination names without uploading or requesting interactive sign-in; source contents, sizes and write permission are checked during delivery. Each upload/verification operation has a two-minute limit.

Existing names always skip at OneDrive upload destinations, including folders that occupy a matching name. This applies regardless of source date or replacement checkboxes. No old file is overwritten and no alternate filename is created. New files upload in chunks with Microsoft's create-only conflict rule, then the application reads them back and verifies their SHA-256 contents, size, folder and stable version. OneDrive assigns the destination modification time. Automatic newer-file replacement is still a later stage for uploads; it remains available for OneDrive sources to previously supported destinations.

The chosen destination is saved as a location and can be selected again through **Add Saved Location**. Use **Setup → OneDrive Destination Accounts...** to select or reconnect an upload account/folder. Saved Locations lists upload destinations for reference and deletion; Delete Location removes the saved reference without deleting cloud files. Choosing a folder later does not silently redirect an existing feed. Removing a saved upload account prevents its feeds from running until that same account is connected again.

Save the feed normally and use its Interval, Daily, Weekly or One Time schedule. In-app/tray and closed-app Background Runs use the same destination and verification rules. Silent sign-in must work; scheduled runs do not open a browser to request consent. A failed or interrupted run disables its schedule for review. After rebuilding, update the Background Runs application path if needed. Remain signed into Windows with the computer awake.

Any upload or verification failure retains the source. An uncertain final result may mean the new file reached OneDrive: review the final file before retrying. There is no automatic upload retry or final-file deletion. Incomplete upload sessions expire automatically. If source archiving is enabled, a file moves into Used Files only after every selected destination reports Copied or Replaced; a OneDrive skip or failure retains it. Remote source originals remain on their servers.

An interrupted upload can leave a zero-byte item reported by Microsoft even when no completed file is visible in the browser. Preview reports a matching item's name, size, account, current folder name and item ID. There is no OneDrive delivery-record reset. Inspect the matching cloud item before removing an empty test copy, or use a fresh test filename. A blocked upload address reports only its server name, not its private upload link. Successful uploads are read back and verified before they report Copied.

Understand OneDrive timestamps and results

The comparison is the OneDrive source modification time against the destination modification time, not filename alone or different contents. Remote comparisons use whole UTC seconds. An edit must have a timestamp later than the destination's reported server time before it can replace that file. Wait for OneDrive synchronization before Preview or Run Now.

Local copies preserve source time. SFTP, FTP/FTPS and HTTPS may retain their own usable server time, shown in result Details and used for later comparisons. A server may also change time when renaming a file. FTP/FTPS accepts that change after verifying final contents, size, regular-file type and a UTC timestamp at least as new as the source. Missing/unusable metadata or failed verification reports Failed even if a file reached the destination; inspect it before retrying.

SFTP requires a saved verified host fingerprint. FTP/FTPS requires the saved checked-upload policy and UTC metadata support; FTPS also requires a trusted TLS certificate and encrypted data connection. HTTPS requires compatible conditional uploads, metadata and read-back access. See each protocol's OneDrive topic for details. Protected SFTP/FTP/FTPS replacement can leave a recovery file after failure or cleanup denial; its reported path identifies what to inspect. Never treat a failed or uncertain publication as permission to delete the final file.

Find text in Help

Press **Ctrl+F** while Help is open to focus the Find box, then type a word or phrase. Search ignores letter case and looks through the full guide, including text in tables and formatted paragraphs. The current match is highlighted in yellow, scrolled into view, and its position is shown as, for example, **1 of 12**.

Use **Next** or **Enter/F3** for the next match, and **Previous** or **Shift+F3** for the previous match. Navigation wraps between the first and last match. **No matches** means no occurrence was found; clearing the box removes the selection. Search does not edit the guide or affect feeds, schedules or files.

Main window menus

File contains Hide To Tray and Exit. **Setup** contains Saved Locations, Outlook Mailbox Accounts, OneDrive Source Accounts, OneDrive Destination Accounts and Background Runs. **Feeds** contains Save Feed, Saved Feeds and Delivery History. **Help** contains User Guide (F1), License Agreement, Privacy Policy and About, which shows the installed version. These menu entries open the existing windows and actions. To configure Outlook, open Setup → Outlook Mailbox Accounts and choose a mailbox folder. To configure OneDrive sources, use Setup → OneDrive Source Accounts. Source selection uses Browse or Use Saved; Use Saved includes configured Outlook and OneDrive source folders alongside folder/server sources. Outlook is never offered as a destination. Hide To Tray keeps the application running; Exit closes it. Background Runs separately permits enabled schedules while the app is closed. Resaving a feed still disables its schedule until reviewed and re-enabled.

Saved Locations window

Saved Locations keeps your saved folder and server list on the left, showing each name and connection type. Select a row to edit its settings on the right, or click **New Location**. The heading identifies the location being edited. Location Details, Connection, Sign-In and FTP upload options group the applicable fields; unrelated fields stay hidden. Scroll the settings while the status and three equal-sized action buttons stay visible at the bottom. **Save Location** saves without testing; **Save And Test** saves and checks; **Test Saved Connection** uses the selected saved settings, ignoring unsaved edits. Saving clears the fields as before. Use **Setup → Outlook Mailbox Accounts...** on the main window to manage your saved personal/work mailbox connections.

Connect a OneDrive account

Choose **Setup → OneDrive Source Accounts....** You can save personal or work/school OneDrive connections, choose folders, inspect files and deliver OneDrive files to existing local folders or saved SFTP, FTP, FTPS and compatible HTTPS destinations. OneDrive feeds support Preview, Run Now, History, saved feeds, all four schedule types and Background Runs. To upload files into OneDrive, use Setup → OneDrive Destination Accounts. Its OneDrive Destination Accounts window uses separate upload connections. Your existing Outlook connections and feeds remain separate.

Open OneDrive on the web first and confirm the account has an available drive. Work accounts need an appropriate OneDrive entitlement and may require company approval. Microsoft documents that a drive-metadata request can automatically initialize an unprovisioned drive for a licensed user. The app does not upload, download, rename or delete file contents during this check. See [Microsoft's drive API documentation](#).

1. Expand **Add A OneDrive Account**, choose **Personal OneDrive** or **Work / School OneDrive**, then click **Sign In And Save**.

2. Select the intended account on Microsoft's page and approve delegated **Files.Read** access. The app supplies the Application ID automatically. No client secret, Mail.Send, Files.ReadWrite or application-level permissions are requested for this read-only OneDrive source feature. An existing Outlook sign-in does not replace this OneDrive sign-in step.
3. After a successful drive check, the account is saved and selected. The status shows its address, drive name and personal/business drive type. A failed check preserves existing accounts.
4. **Test Saved Connection** checks the selected saved account silently. If Microsoft needs renewed consent or sign-in, use **Reconnect Selected Account**. Reconnect uses the saved registration and account type, refusing a different account/directory before committing the sign-in.
5. Add more accounts as needed. Switch using **Saved OneDrive accounts**; adding or reconnecting the same identity does not create a duplicate. Connections persist after restart.
6. Under **Connection Management**, **Disconnect Selected Account** removes only that OneDrive connection after confirmation. Other OneDrive accounts, all files, Outlook connections, email attachment records and saved feeds remain. Microsoft/browser consent is not revoked. Encrypted sign-in storage is separate from Outlook's; accounts using the same Application ID within OneDrive share that OneDrive cache until its last connection is disconnected.

The check performs one authenticated GET to the selected account's default drive on Microsoft Graph, without following redirects. It reads at most 64 KiB of metadata and has a 30-second request deadline. Test Saved Connection does not enumerate folders, download files, prove upload permissions, or test SharePoint libraries. Choose OneDrive Folder and Inspect Files are separate metadata-only actions. The current personal and work options expect personal and business drives respectively. A sign-in operation can be cancelled; closing while busy cancels before the window closes. Errors are reported without displaying access tokens or raw server response bodies.

Connection verification: keep test schedules and Background Runs off while rebuilding; open Setup → OneDrive Source Accounts, add/test the personal account, close/reopen and restart/test again. Reconnect and confirm no duplicate appears. Add the work account, switch between both and test each. Open a saved Outlook mailbox afterward and test/inspect it to confirm its connection remains available. No new test files or folders are needed for connection checks.

Choose a OneDrive folder and inspect files

In **Setup → OneDrive Source Accounts**, select a saved account and click **Choose OneDrive Folder....** The browser starts at the root of that account's default drive. Select a folder and click **Open Folder** to enter it, then repeat for deeper folders. **Back** returns one level. **Use This Folder** saves the level currently displayed, including the root; it does not save a merely highlighted child. **Cancel** keeps the previous choice. Only ordinary folders in your own drive are shown; shared shortcuts, special packages such as notebooks and deleted entries are excluded.

The saved path appears under the account actions. Each account retains its own choice across switching, reconnecting and restart. Selection is bound to the account, company directory, drive ID and folder ID. A display path can become stale after a rename or move; the folder ID is used to check access. If the drive changes or the folder becomes unavailable, select it again. Choosing a folder does not itself create a feed; use Use As Source, select destinations and Save Feed when ready.

Inspect Files... lists ordinary files directly inside that saved folder, showing original name, size in bytes and modification time in UTC. Subfolders are counted but not inspected recursively. Shared shortcuts, packages and deleted entries are excluded. The table is read-only; select rows and press Ctrl+C to copy with headers. Close and reopen inspection to refresh. Empty folders produce a successful empty list. File contents are not read, and no remote files are created, renamed or deleted.

Folder operations use silent saved sign-in and the existing Files.Read permission. They have a two-minute deadline, at most 5,000 entries and 40 listing pages per folder, 2 MiB per response and 8 MiB total listing metadata. Continuation links

must stay on the same Microsoft Graph folder endpoint with the same selected metadata. Errors, changed accounts, unsupported folders or exceeded limits return no partial inspection and retain the saved choice. The folder is checked again before saving. See [Microsoft's folder-listing API](#).

Live verification: use an existing manageable folder with ordinary files and a nested folder where available. Browse into it and Use This Folder; compare inspection names/sizes with OneDrive web and verify that nested-folder files are absent. Select the nested folder separately and inspect it, then return to the original folder. Try Cancel after browsing elsewhere and confirm the saved choice remains. Switch personal/work accounts, choose independent folders, reconnect, reopen/restart and confirm each choice persists. No new test folders or files are required if suitable existing ones are available; these inspection checks do not transfer files.

Connect an Outlook online mailbox

The current version saves multiple personal Outlook.com/Hotmail and work/school Microsoft 365 online mailboxes, verifies Inbox access, saves a mailbox-folder choice, inspects attachment metadata and delivers ordinary attachments to local/network folders and saved SFTP, FTP, FTPS or compatible HTTPS destinations. Open **Setup → Outlook Mailbox Accounts...** to configure a connection, or **Use Saved...** in the main window's Source folder area to select it for a feed. Work mailboxes must be hosted in Microsoft 365/Exchange Online; using the Outlook desktop program with another provider does not make that mailbox compatible. This connection uses Microsoft sign-in directly; the Outlook desktop application does not need to run. Mailbox feeds support Preview, Run Now, History and the shared scheduler. Choose a connection in Saved mailboxes; each feed keeps its own mailbox and folder.

Keep several saved mailboxes

Use **Saved mailboxes** to choose an existing personal or work connection. Its account type is read-only and its selected folder returns automatically. To add another existing email account, click **Add Mailbox....** Under **Add A Microsoft Mailbox**, choose Personal or Work / School and click **Sign In And Save**. The app supplies its Application ID automatically. This connects an existing mailbox; it does not create a new email address. Select the intended account on Microsoft's page. Successful verification adds it to the list. **Reconnect Selected Mailbox** uses the saved type and Application ID, ignoring Add settings, and retains the folder without adding a duplicate. Choosing a different account during reconnect is refused without changing the saved connection. No extra Microsoft permissions are requested.

The mailbox window separates saved-account actions, Add A Microsoft Mailbox and direct-download actions. Advanced Microsoft Setup is inside the Add section. Action buttons have equal sizes and aligned columns. Saved Mailboxes, Test Saved Connection, Choose Mailbox Folder, Inspect Attachments, the selected folder and Status are grouped together at the top. The three mailbox action buttons share a row, wrapping when the window is narrow. Close / Cancel Operation stays at the bottom while the settings scroll. Disconnect and Reset Attachment Records are under **Connection And Delivery Record Management**. Test Saved Connection explicitly reports the main Inbox count; **Inspect Attachments** reads the selected attachment folder instead. Each saved feed keeps its own mailbox and folder, regardless of the mailbox currently selected in this window. The same applies to scheduled and closed-app runs. **Disconnect** asks for confirmation and removes only the selected connection; feeds needing it fail until the same account is reconnected. Other connections and attachment records remain. **Reset Attachment Records** still applies to all mailboxes and destinations for this Windows user.

To verify multiple mailboxes, keep Background Runs and test schedules off while rebuilding. Check the existing connection and folder, add the other account, select its existing attachment folder, then switch between both saved connections and test/inspect each. Reconnect one account and confirm no duplicate appears. Restart and confirm both remain. Finally, Preview and Run Now an existing feed for one account while the other is selected: it should use the feed's account and report normal copied/already-delivered results. Reuse existing emails and destination folders; a reset is unnecessary for verifying account selection.

One-time Microsoft application setup

Microsoft requires an application registration maintained by the publisher. This app supplies the current public **Application (client) ID** automatically, so ordinary users choose an account type and sign in. **Advanced Microsoft Setup** permits a development or organization-specific override when adding an account. Normally leave it unchanged: an incorrect ID can prevent sign-in, and changing it does not retarget existing accounts or feeds. Commercial release still requires reviewing registration ownership, publisher verification, consent information and privacy/support details; supplying the ID does not complete those release tasks. The registration instructions below are for the publisher or someone intentionally using their own registration. Registration requires access to a Microsoft Entra tenant and permission to register applications; a personal mailbox alone does not necessarily provide that access.

1. Open <https://entra.microsoft.com/> and sign in with an account that can manage an application registration. Microsoft's prerequisites are at <https://learn.microsoft.com/en-us/graph/auth-register-app-v2>.
2. Open **Identity** → **Applications** → **App registrations** → **New registration**. Name it **Automated File Routing**.
3. To use both mailbox types, choose **Accounts in any organizational directory and personal Microsoft accounts**. A personal-only registration can still be used with the Personal option, but cannot sign in a work account.
4. Register, then copy **Application (client) ID** from Overview. This is a public identifier, not the Object ID, mailbox password, access token or client secret.
5. Under **Authentication** → **Add a platform**, choose **Mobile and desktop applications** and add <http://localhost> as a redirect URI. Configure/save it. This is Microsoft's local browser sign-in callback; it does not require installing or running a test web server.
6. Under **API permissions** → **Add a permission** → **Microsoft Graph** → **Delegated permissions**, add **Mail.Read**. It supports attachment reading; Mail.ReadBasic cannot read attachments. Do not add Mail.Send or Mail.ReadWrite for this feature. No client secret is used by this desktop connection.

The app uses Microsoft's MSAL library for browser sign-in and saved sign-in renewal. Complete consent on Microsoft's page. It requests read-only Mail.Read access (plus the library's normal sign-in/session scopes), not permission to send or change mail. Consent may be revoked later through the Microsoft account. If consent, account registration or local browser callback is refused, the window reports a failure and retains the previous saved connection.

First mailbox connection check

1. Disable Background Runs before rebuilding. Rebuild, open the application and choose Saved Locations → Outlook Mailboxes.
2. Click **Add Mailbox....** The Application ID is supplied automatically; ordinary users need not open Advanced Microsoft Setup.
3. Choose **Personal Outlook.com / Hotmail**, then click **Sign In And Save**. Choose the intended personal account on Microsoft's browser page and consent to reading mail.
4. Return to the app. It should report the verified/saved mailbox, Inbox name, message count and unread count. No message bodies or attachments are downloaded; no messages are marked read, sent, moved or deleted.
5. Click **Test Saved Connection**. It should check the saved mailbox without an interactive browser prompt. If Microsoft requires renewed consent or sign-in, use Sign In And Save again; Test does not automatically launch a browser.
6. Close and reopen the Outlook Mailbox window, then restart the application and reopen it. Confirm the saved mailbox and Application ID remain and Test Saved Connection still works.

7. Close the Outlook desktop program if it is running and test again. Online access should still work. Check mailbox unread state/messages remain unchanged.
8. Optional disconnect test: click **Disconnect**. Confirm only the selected connection is removed (Test Saved Connection is disabled if none remain); no messages were removed. Sign In And Save again to leave the test mailbox connected for the next stage.

Sign In And Save adds or refreshes the matching account only after the Inbox check succeeds. Other saved mailboxes remain available in the Saved mailboxes selector. Tests ignore unsaved Application ID edits. **Cancel Operation** stops a pending check or browser sign-in; closing while busy cancels the operation before closing. The results wrap and can be copied. Status failures, malformed metadata or excessive responses do not overwrite the previous connection.

Public connection/account identifiers are stored in the application's OutlookMailbox data folder. Sign-in tokens are stored separately in an encrypted cache protected for the current Windows user and bound to this application's registered ID; they are not stored in feed/location JSON or shown in the window. This cache is larger than Windows Credential Manager's per-item limit, so it uses Windows user encryption instead. Disconnect removes only the selected saved connection. When other saved mailboxes use the same Application ID, the shared encrypted sign-in cache is kept for them; it may retain sign-in information for the removed account. After the final connection for that Application ID is disconnected, its cache is deleted. Disconnect does not revoke Microsoft consent or sign the browser out. Another Windows account or computer cannot simply reuse the encrypted cache. Do not share sign-in cache files or tokens.

Add work/school mailbox support

Reuse the existing Azure account and application registration where possible. In Microsoft Entra → App registrations → Automated File Routing → Authentication (Preview) → Supported accounts, change support to **Accounts in any organizational directory and personal Microsoft accounts** and save. If the portal does not offer the change there, use the registration's Manifest editor: preserve/download a copy first, then change only `signInAudience` from `PersonalMicrosoftAccount` to `AzureADandPersonalMicrosoftAccount` and save. Microsoft's instructions are at <https://learn.microsoft.com/en-us/entra/identity-platform/howto-modify-supported-accounts>. Keep the Application ID, desktop `http://localhost` redirect and delegated `Mail.Read` permission. No new client secret or OneDrive file permissions are needed.

In the application, select **Work / School Microsoft 365**, then Sign In And Save with the work account. A company may require administrator approval, publisher verification or additional sign-in conditions; the app cannot bypass those policies. A work user needs an available Exchange Online mailbox. The Azure directory created for application registration alone does not supply a work mailbox.

Multiple mailbox connections can be saved. Successful sign-in to a new mailbox adds and selects it; an unsuccessful Inbox check retains existing connections. Re-signing into the same account refreshes it without adding a duplicate or losing its selected folder. Test Saved Connection uses the saved type/account/company directory, ignoring unsaved dropdown edits. Older personal connections load as Personal automatically. The connection list uses schema 4 and imports earlier single-connection settings. Earlier builds refuse the new format; use this build for background runs too. Work saved tests remain pinned to the verified company directory and account. Tests do not launch interactive sign-in automatically.

Work-support and pending connection checks:

1. Keep Background Runs OFF, rebuild, and open Saved Locations → Outlook Mailboxes. Confirm the existing personal mailbox remains saved with Personal selected; Test Saved Connection should succeed.
2. Close and reopen the mailbox window and test again.
3. Exit and restart the application, reopen the mailbox window, and test again.

4. Close Outlook desktop, test again, and independently check in Outlook web that the test message is still unread. Other clients/new incoming mail can change counts; the check should not mark a message read.
5. Update the existing Microsoft registration to support organizational and personal accounts as described above.
6. Select Work / School Microsoft 365, keep the same Application ID, and Sign In And Save with the intended work mailbox. Confirm its address, Work account type and Inbox metadata. If company approval is required, report the message and leave that work test pending.
7. Test Saved Connection, close/reopen the mailbox window, restart the app and test again. Confirm the work mailbox/type remain saved and no interactive browser appears unless renewed sign-in is required.
8. Select Personal, Sign In And Save with the personal mailbox, and test again. Confirm switching back works and neither mailbox's messages were modified. Leave the desired account connected for the next stage.

Choose a mailbox folder and inspect attachments

In Outlook Mailbox, click **Choose Mailbox Folder...**. The picker reads the saved connection, ignoring unsaved Application ID/account-type edits. It starts at the mailbox's visible top-level folders. Select a row and click **Open Subfolders** to browse deeper, or **Back** to return one level. Click **Use Selected Folder** to save that folder's ID/path locally. Cancel leaves the saved choice unchanged. Hidden folders are excluded; reported subfolder counts can include hidden entries that are not shown.

The chosen folder appears in bold in Outlook Mailbox and persists after restart. **Inspect Attachments...** is enabled once a folder is saved. It shows file attachment names, reported sizes, message subjects, senders and received times in UTC. The app explicitly requests metadata, not message bodies or attachment content. It does not mark messages read, send mail, move/delete messages or create a feed. Inline images, attached messages and linked/cloud-reference attachments are excluded. Messages containing only inline images may not appear in Microsoft's hasAttachments filter. Duplicate filenames remain separate rows from their respective messages; no filename conflict is resolved at this inspection stage. Select rows and press Ctrl+C to copy; long cells wrap, and copying preserves the full value.

Only the selected folder is inspected; its subfolders are separate choices. Close the inspection window and click Inspect Attachments again to refresh. Empty folders produce a successful empty table. Inspection is a snapshot rather than a mailbox lock. If items move or disappear, access is refused, metadata is malformed, or limits are exceeded, the operation stops without presenting partial results or changing the saved folder. Limits are 1,000 folders per browsed level, 200 messages with attachments, 2,000 attachment entries, 25 pages per listing, 300 requests and 8 MiB total metadata per operation, with 2 MiB per response and a two-minute inspection deadline. Choose a smaller dedicated folder if a limit is reached. Pagination stays on the same Microsoft Graph endpoint with the same metadata selection/filter; unexpected links or redirects are refused.

The saved folder is bound to the mailbox account and work directory, not just its displayed address. Re-signing into the same account retains the folder. Each saved mailbox retains its own folder. A newly added mailbox needs its first folder selection. Choosing another saved mailbox restores its folder without changing existing feeds. Folder IDs are used for access; the saved human-readable path can become stale after a rename. Inspection verifies the folder and displays its current name. The connection list imports earlier schema-1/2/3 connections and saves schema 4; older builds cannot read the new format.

First live inspection test (personal mailbox first):

1. Keep Background Runs OFF, rebuild and open Saved Locations → Outlook Mailboxes. Test the saved connection.
2. Create `mail-test-one.txt` containing First attachment and `mail-test-two.txt` containing Second attachment. Using Outlook's website, send yourself a disposable email with subject Attachment Test and both files as ordinary attachments, not OneDrive links.

3. Under Inbox, create a mailbox folder named AttachmentTest. Move the received test email into it, then mark it unread. Create a Nested folder inside AttachmentTest and put a different disposable email with an attachment there, to check subfolder exclusion.
4. Click Choose Mailbox Folder. Select Inbox → Open Subfolders, select AttachmentTest → Use Selected Folder. Confirm the saved path names Inbox/AttachmentTest.
5. Click Inspect Attachments. Confirm both filenames, reported sizes, subject/sender and received time. The attachment from Nested should not appear.
6. Copy a row with Ctrl+C and paste it into a text window. Close inspection, open the folder picker, select another folder and Cancel. Confirm the saved folder is still AttachmentTest.
7. Restart the app and reopen Outlook Mailbox. Confirm the folder remains selected and inspecting again shows the same test attachments.
8. In Outlook web, confirm the test email remains unread and its attachments/messages are unchanged. For the pending desktop-closed connection check, close Outlook desktop if running and Test Saved Connection again.
9. Send yourself another disposable email containing the same mail-test-one.txt filename, move it to AttachmentTest and mark it unread. Reinspect: the filename should appear in two separate rows with their respective message subjects.
10. Create an empty mailbox folder in Outlook web. Choose it and inspect; expect zero attachment rows. Then reselect AttachmentTest for the next stage. Work-mailbox inspection will be checked separately using its own test folder.

Download mailbox attachments to a local folder

After choosing a mailbox folder, click **Download To Folder...** and choose an existing destination folder. Accepting the folder starts delivery of all ordinary file attachments in the selected mailbox folder. Cancel in the folder picker starts nothing. Only the selected mailbox folder is read; nested folders, inline images, attached messages and cloud-reference attachments are excluded, just as in inspection. Outlook messages stay unread/read as they were; no mail is sent, moved or deleted. This shortcut is separate from main-window feeds and does not create a History run or use replacement settings or scheduling. It shares delivery tracking with mailbox feeds: an attachment already downloaded to the same destination is skipped by either path. Copy the shortcut's result text in the mailbox window to keep or report it.

Attachments keep their original filenames when available: mail-test-one.txt and mail-test-two.txt stay that way. If a name is already used by another file, folder, attachment in this run or saved delivery record, a short suffix is added before the extension, for example report-a1b2c3d4.csv. Further collisions receive a counter. There is no added mail-prefix. Unsafe Windows filename characters/device names are adjusted and very long names shortened. File contents are not edited or opened.

Each chosen output name is saved with that attachment's delivery record for that destination. Previously downloaded files keep their old names, including the longer prefixed names from the first download build; the update does not rename or resend them. Pending deliveries retain their assigned names for recovery. Names at different destinations can differ according to availability and delivery order. Microsoft immutable message IDs keep tracking stable when a message moves between folders in the same mailbox; moving to another mailbox/archive or recreating a message may create a new identity. The actual streamed byte count is reported separately from Microsoft's reported attachment size.

The application records successful delivery separately for each attachment and destination folder. Repeating delivery to the same destination reports **Already delivered** without downloading that attachment again. Choosing a different destination copies it there independently. Restarting or disconnecting/reconnecting the same mailbox keeps delivery records. Deleting or changing an already-delivered destination file does not reset processing or make the app resend it. Use the explicit **Reset Attachment Records...** action below if you intentionally want to download the same attachments

again. Destination paths are compared without case differences; different drive/share aliases for the same actual folder can be treated as separate destinations. Prefer one consistent destination path.

Pending attachments are downloaded to a temporary staging folder and their metadata is rechecked before any new destination file is published. Limits are 25 MiB per attachment, 250 MiB of pending attachment content per batch, the inspection metadata limits above (including extra verification requests), and a five-minute window-operation deadline. Download redirects, partial/encoded responses, changed/missing metadata and invalid tracking records stop delivery. A smaller dedicated mailbox folder helps keep within limits. Raw downloads use Microsoft's file-attachment \$value endpoint with read-only Mail.Read permission; see [Microsoft's attachment documentation](#).

Each destination copy is staged and checked with a SHA-256 checksum, then renamed without overwriting an existing file. Existing names are avoided when assigning a new filename. If a file appears at that assigned name during delivery, or conflicts with a pending recovery record, delivery stops for review without overwriting it. Completed copies remain if cancellation or an error interrupts publication; the result lists them, and the next run uses saved progress. A pending record can recover a file published just before a stop, after checking its size and checksum. There is no automatic retry. A cleanup warning gives the exact temporary path if staged files could not be removed. Linked destination folders/files are refused.

First live local-download checks (start with your currently connected work mailbox):

1. Turn Background Runs OFF, stop the app and rebuild. Open Saved Locations → Outlook Mailboxes and Test Saved Connection.
2. Choose the work mailbox's Inbox/AttachmentTest folder. Inspect Attachments and confirm the expected ordinary attachments. Keep the test emails unread.
3. Create a new empty local folder called Work Mail Downloads. Click Download To Folder, select that folder, and accept. Confirm the result says Finished and reports copied attachments with zero errors.
4. Open the local files in Notepad (or their normal program) and confirm their contents match the email attachments. Different available filenames stay unchanged, with no added prefix or suffix. If multiple attachments have the same name, the additional copies have a short suffix; file extensions should remain intact. Earlier downloads retain their previous assigned names, so use a new empty local folder to check the updated naming behavior.
5. Repeat Download To Folder using the same destination. Confirm zero copied and the existing attachments reported Already delivered; no extra local files should appear.
6. Send yourself another disposable email containing an ordinary attachment with the same original filename as an existing test attachment but different contents. Move the received email into AttachmentTest. Run Download To Folder to the same local destination. Confirm only the new attachment copies, its name differs, and both versions' contents remain intact.
7. Put a different email/attachment in AttachmentTest/Nested. Repeat delivery and confirm the nested attachment does not copy. In Outlook web, confirm the test messages remain unread and unchanged.
8. Restart our app, reopen Outlook Mailbox and repeat delivery to the original local destination. Confirm the folder choice persists and all previously delivered attachments are skipped.
9. Create another empty local folder called Work Mail Downloads Second. Download to it and confirm all current eligible attachments copy with matching contents. Names are assigned independently based on availability; an empty destination keeps available original names and suffixes additional duplicates. The first destination must remain unchanged.
10. Select an empty mailbox folder and Download To Folder to a new empty local destination. Confirm zero copied and successful Finished status. Reselect AttachmentTest, open Download To Folder and cancel the folder picker; confirm

nothing copies. Select/copy the result text to confirm it is easy to report. Personal-mailbox downloads and feeds/scheduling will be tested separately.

Reset attachment delivery records and reuse test folders

Open **Saved Locations** → **Outlook Mailboxes...** → **Reset Attachment Records...** This button clears **all attachment delivery records for this Windows user, across all mailboxes and destination folders**, including records for destinations that have since been deleted. It is available without connecting to a mailbox. The warning defaults to **No**; cancelling changes no records. Confirm **Yes** only when you intend to allow old attachments to download again. The reset cannot be undone.

Disable test feed schedules and Background Runs first if you want to control when downloading resumes. An enabled schedule can download those same old attachments on its next run after reset. The button itself only clears local tracking; it does not connect to Microsoft, start a download, recreate destination folders, delete destination files, change email, sign out, alter saved feeds/schedules or erase History. If a destination folder was deleted, recreate it at the same path before using the feed again, or update the feed to another existing folder.

After reset, an empty destination can receive the same attachments under their original available filenames. If earlier copies still exist, new copies receive suffixes. FTP/FTPS retains its saved upload-policy and filename-check race limitation, described below. Normal duplicate prevention resumes as the new successful deliveries are recorded. This lets you reuse the same emails, filenames and destination paths deliberately. Reset shares the delivery lock with manual and scheduled runs, so it waits for an active operation rather than clearing that operation's tracking halfway through. It reports cleared and failed record counts; if any records could not be cleared, the reset is incomplete and those attachments may still skip. Check local file access and retry. Only application receipt files are removed, without visiting destination folders or removing subfolders.

Use Outlook attachments in a saved feed

Email attachment routing does not use Replace. An attachment keeps its original filename when that name is available. If another file already uses the name, the new attachment receives a suffix to preserve both files. Previously delivered attachments are skipped. **Reset Attachment Records** allows those same attachments to download again; it does not enable replacement. This preservation rule applies to local/network, SFTP, FTP/FTPS and HTTPS email destinations. FTP/FTPS still uses its saved upload-protection policy and filename-check race warning; HTTPS requires the endpoint to honor conditional create-only uploads, explained below.

1. Open **Setup** → **Outlook Mailbox Accounts** to test the account and choose a mailbox folder. Use **Use As Source** there, or select the configured folder through **Use Saved...** beside the main source box. Cancelling the source picker leaves the main source unchanged; account/folder choices explicitly saved inside the mailbox window remain saved there.
2. Add local/network folders or use **Add Saved Location** to select SFTP, FTP, FTPS or compatible HTTPS destinations. They can be combined in one feed. For network runs, prefer a full share path over a mapped drive. Replacement and source archiving are off and disabled for Outlook sources: available filenames remain original, duplicates receive suffixes, previously delivered attachments are skipped and emails remain unchanged. FTP/FTPS requires its saved upload permission policy; HTTPS requires conditional-create support.
3. **Preview** reads attachment metadata and delivery records without downloading contents or writing destination files. It shows planned output filenames and **Would copy** or **Skipped / Already delivered**. Pending interrupted deliveries can preview as Would copy but be recovered after checksum verification. Preview does not prove download permission or contents; names are checked again at run time.
4. **Run Now** delivers using the same checked local-download path. Results and **History** include the readable mailbox/folder, saved attachment filenames, copied/skipped/failed outcomes and per-destination details. Empty mailbox folders produce successful empty runs. Completed copies and tracking survive partial failures; inspect the

failed details before retrying.

5. **Save Feed...** stores the account/application/directory identifiers and chosen folder, together with destinations. Tokens remain in encrypted sign-in storage, never in the feed. Saving can clear current selections as usual. Load through **Saved Feeds** → **Use Feed**. Re-saving an existing feed disables its schedule; review and re-enable it in Saved Feeds.

A feed captures its own mailbox folder: later browsing to another folder in the mailbox window does not change that feed. Folder IDs are used, so the displayed saved path can become stale after a rename. Each feed resolves its own separately saved mailbox, even when another mailbox is selected in the window. Disconnecting its connection makes that feed fail safely until the same account is connected again. Other mailbox feeds remain available. Changing application registration or work directory requires re-establishing the matching connection.

Enable a mailbox feed's interval/calendar schedule in **Saved Feeds** → **Schedule**, with automatic connection retries set to **0**. The in-app scheduler runs while the application is open or in the tray, after closing Saved Feeds/configuration windows. **Background Runs** additionally permits enabled feeds to run with the app fully closed while Windows is signed in and the computer is awake. Mailbox runs use silent saved sign-in; they never open an interactive sign-in browser. A sign-in/access/download/tracking/cleanup failure or History-save failure disables the affected schedule for review. Reconnect through Setup → Outlook Mailbox Accounts when needed, then explicitly re-enable the schedule. The shared runner has automated coverage; live mailbox scheduling and closed-app tests are separate from the manual feed checks below.

Settings now use schema 10 and saved-feed documents schema 12, with older supported formats still readable. Older application builds reject the new documents rather than silently treating a mailbox as a local source. Returning to an older build also requires compatible saved settings/feeds.

First live mailbox-feed checks (use the currently connected personal mailbox; keep Background Runs OFF):

1. Stop the app, rebuild and reopen it. Clear current selections so the test has no leftover destinations. Click Setup → Outlook Mailbox Accounts, test the saved personal account, select Inbox/AttachmentTest, then Use As Source. Confirm its address/folder appear in the main source box.
2. Add a new empty local destination named Outlook Feed Destination. Confirm archiving/replacement options are off/disabled and the explanatory text describes Outlook duplicates and processed skipping.
3. Click Preview. Confirm eligible attachments show Would copy with available original names; check the destination is still empty. Nested-folder attachments should not appear.
4. Save as Outlook Feed Test. If saving clears selections, use Saved Feeds → select Outlook Feed Test → Use Feed. Confirm the mailbox/folder and destination restore correctly.
5. Run Now. Confirm copied results, matching file contents and a History entry identifying the Outlook source and destination filenames. Emails should remain unread and unchanged.
6. Preview and Run Now again. Confirm Already delivered/Skipped and zero additional copies.
7. Send a new disposable email with an ordinary attachment sharing an existing filename but containing different text. Move it to AttachmentTest. Preview and Run Now: only the new attachment should copy, with a suffix, preserving both contents.
8. Through Saved Locations → Outlook Mailboxes, choose another mailbox folder and close the window. Preview the loaded feed: it should still read its original AttachmentTest folder. Do not switch accounts for this check.
9. Restart our app, load Outlook Feed Test from Saved Feeds and Preview. Confirm the captured source and delivery tracking persist, with existing attachments skipped.

10. Add a second new empty local destination. Re-save/replace Outlook Feed Test and load it again if selections clear. Preview/Run Now should skip completed attachments at the first destination and copy all eligible attachments to the second. Confirm both destinations' contents and separate History results. Leave the test feed unscheduled for the next scheduling stage.

Deliver Outlook attachments to SFTP

Select an existing working **SFTP** destination using **Add Saved Location** after choosing the mailbox source. The saved location needs valid credentials, the verified SHA256 server fingerprint, and a writable remote folder. This email-delivery stage also requires directory listing and read-back permission: filenames are planned around every existing entry, and copied contents are checked by reading them back. A successful connection test alone does not prove upload, rename or read-back permission.

Preview reads mailbox metadata and SFTP directory names, with no attachment downloads or remote writes. Available original filenames remain unchanged; duplicate names are suffixed. The application reserves existing files, directories and saved output names when planning. Case-conflicting or invalid directory listings are refused. Only the selected mailbox folder is used, without subfolders. The file-size, metadata, batch and timeout limits from mailbox downloading still apply.

Run Now downloads and rechecks all pending attachments for each SFTP destination before starting its uploads. It creates each remote temporary upload exclusively, verifies the temporary contents, publishes with a no-overwrite rename, then verifies the published contents before recording completion. It never uses SFTP replacement or deletes a final destination file. A filename appearing during publication stops that delivery for review. Only a temporary upload created by this operation can be cleaned up automatically. Cleanup failures identify the remaining temporary path in the results.

Delivery records identify the mailbox attachment and the SFTP server/port/account/folder/verified identity. Renaming a location or creating another saved profile for that same endpoint does not by itself cause a repeat upload. Changes to connection endpoint, account, fingerprint or folder can create a separate destination identity. Path case and relative/absolute paths remain distinct; use a consistent saved path, because server aliases cannot all be identified. Already-delivered attachments skip both the mailbox download and the upload. Local and SFTP delivery progress is tracked independently, including mixed feeds.

If a connection fails near publication, a pending record retains the intended filename, size and checksum. A later run can recover a matching published file after reading it back, without uploading it again. A missing target permits resuming the checked delivery; different or unreadable contents stop for review. No automatic upload retry is performed. Inspect remote files before resetting records after an uncertain result. **Reset Attachment Records** clears SFTP records as well as local records: existing remote copies remain and repeated delivery receives suffixes. If you manually empty the same remote folder and then reset, the same original filenames can be reused.

SFTP mailbox feeds use the same main-window/History and shared scheduled/background paths as local mailbox feeds. A failure disables an affected scheduled feed for review. Live SFTP scheduling remains a separate test after manual SFTP delivery passes. First live checks, using the existing emails, filenames and folders:

1. Keep the test feed schedule and Background Runs OFF. Stop the app, rebuild and reopen it.
2. Verify a currently working SFTP test location in Saved Locations using its saved connection test. Confirm the target folder and access through FileZilla. Update an expired test account before continuing.
3. Clear main selections, Setup → Outlook Mailbox Accounts to select the existing AttachmentTest source, then Add Saved Location to add the SFTP destination. Archiving/replacement should remain off and disabled.
4. Preview. Confirm Would copy and the planned names; existing remote names should cause suffixes. Refresh FileZilla and confirm Preview uploaded nothing.

5. Save as Outlook SFTP Test, then restore through Saved Feeds → Use Feed if saving clears selections. Confirm source and SFTP destination restore.
6. Run Now. Confirm Copied results and SFTP History. Download the resulting remote files through FileZilla and compare their contents with the email attachments. Existing remote files must stay unchanged.
7. Preview and Run Now again. Expect Already delivered/Skipped, with no additional remote files.
8. With schedules still off, Reset Attachment Records and run the same feed using the same emails again. Existing remote copies should remain and additional copies receive suffixes.
9. Add the existing local test destination (recreate its same path if needed), re-save the feed and run it. Confirm SFTP skips completed attachments while local copies succeed with matching contents and no overwrites. Check per-destination History.
10. Restart, reload the feed and Preview. Confirm both destinations report Already delivered and the emails remain unread/unchanged. Leave the feed unscheduled until the next test stage.

Deliver Outlook attachments to FTP or FTPS

Use **Add Saved Location** to select a working FTP or FTPS destination after selecting a mailbox source. Reuse the existing saved protocol, security mode, credentials, passive/active setting and remote folder. FTPS retains trusted-certificate validation, encrypted control/data connections and the configured explicit or implicit mode; it does not fall back to plaintext FTP. Plain FTP remains unencrypted. The account needs directory listing, upload, binary read-back, size reporting and rename permission. Connection testing alone does not establish these permissions.

Mailbox feeds honor the existing Saved Locations upload-policy checkboxes. With both unchecked, Preview and Run refuse uploads. **Allow uploads with filename checks** permits the existing checked-upload behavior. **Administrator confirmed: this account rejects overwriting files during both uploads and renames** may be used only when the server provider confirms that policy; a successful connection test does not verify it. No email Replace option is provided, even when checked uploads are allowed.

FTP/FTPS does not provide a portable exclusive-create/no-overwrite-rename guarantee. The application reserves existing names, uses suffixed duplicates and checks again before upload and rename. Another program can still create the same name in the small gap between checking and publication; an overwriting server could then replace it. The saved filename-check consent and its warning therefore still apply to email delivery. A server-enforced no-overwrite policy closes that race. Preview and copied-result details distinguish the two saved policies. Resetting attachment records does not remove this limitation or change the saved policy.

Preview reads mailbox metadata and directory names only. Run stages and rechecks the pending attachments before starting uploads, uses binary transfers, verifies temporary contents through bounded read-back checksums, checks final-name availability again, then renames and verifies published contents before saving completion. It never requests replacement or deletes a final destination file. Completed records skip later downloads/uploads. Endpoint/account/folder/protocol/security-mode changes identify separate destinations; changing only the saved profile name/ID does not. Local, SFTP, FTP and FTPS progress remains independent.

Pending receipts recover a matching published file after an interrupted rename; mismatched or unreadable contents stop for review. No automatic upload retry occurs. If an upload throws or the server does not confirm it, the result reports a possible partial temporary path and does not automatically delete that uncertain file. Confirmed owned temporary uploads may be cleaned after later failures; a cleanup error identifies the path. Review uncertain remote results before retrying or resetting. Reset includes FTP/FTPS records: existing copies remain and new copies receive suffixes; an emptied destination can reuse original names.

Both protocols use shared History and scheduling, with failed scheduled runs disabled for review. First live manual checks (repeat for FTP and FTPS using the same emails/files/folders; keep schedules and Background Runs OFF):

1. Stop the application, rebuild and reopen it.
2. Verify a currently working FTP or FTPS saved location using its connection test and FileZilla. Keep the upload-policy choice previously tested for that account; do not assert an administrator policy without provider confirmation.
3. Clear selections, Setup → Outlook Mailbox Accounts to choose the existing AttachmentTest source, then Add Saved Location to select this FTP/FTPS destination. Replacement and archiving remain off/disabled.
4. Preview. Confirm planned filenames/duplicate suffixes and the saved policy explanation. Refresh FileZilla and confirm no upload occurred.
5. Save as Outlook FTP Test or Outlook FTPS Test; restore through Saved Feeds → Use Feed if selections clear.
6. Run Now. Confirm Copied results and matching contents through a fresh FileZilla download. Existing remote files should remain intact. Check History and its protocol label.
7. Preview/Run again. Confirm Already delivered/Skipped and no additional remote copies.
8. Reset Attachment Records with schedules off and run using the same emails. Confirm existing remote copies remain and additional copies have suffixes.
9. Add the existing local test destination, re-save and run. Confirm FTP/FTPS skips completed attachments while local copies succeed, with separate History results.
10. Restart, restore the feed and Preview. Confirm saved source/destinations and progress persist and emails remain unchanged. Leave the feed unscheduled. Repeat this sequence for the other protocol; scheduling tests follow separately.

Deliver Outlook attachments to HTTPS

Choose a working saved **HTTPS** destination through **Add Saved Location** after choosing the mailbox source. Use the actual upload folder URL, not a browser login/portal page. This is raw-file delivery to endpoints supporting individual-file **HEAD, PUT with If-None-Match: *** and **GET read-back**, using the existing saved None/Basic/Bearer authentication. It does not implement website upload forms or company-specific login/POST adapters. HTTPS connection testing verifies only a HEAD response and authentication, not upload/read-back permission or conditional-create compliance.

Preview reads mailbox metadata and probes candidate filenames with HEAD; it does not download attachments or existing remote contents, upload or delete anything. Available original names are kept and occupied names receive suffixes. It does not require a WebDAV directory listing or file ETag for create-only delivery. Name checks are capped at 100 candidates per attachment and 500 HEAD requests per destination operation, in addition to the existing mailbox limits and five-minute operation deadline. Endpoints returning success for every nonexistent filename cannot be safely planned and will hit the bounded check limit.

Run downloads and rechecks pending attachments before uploading to that HTTPS destination. Every PUT is conditional create-only (If-None-Match: *), with no If-Match, unconditional PUT, replacement selection or remote deletion. **The endpoint must honor that condition** to prevent overwriting a file created after Preview/planning. A 412 precondition refusal stops delivery for review without an automatic suffix retry. After a successful upload response, every file is read back and verified against its byte count and SHA-256 checksum before recording completion. Missing HEAD sizes/ETags are acceptable; missing/refused GET access, changed/short/excess/encoded/partial contents and redirects fail verification. Redirects are never followed and credentials are not forwarded to another URL.

Delivery records distinguish endpoint URL, saved authentication/username/credential reference and mailbox attachment. Renaming a saved profile or changing only its ID does not resend to the same endpoint; different folder URLs or

credential references identify separate destinations. Consistently use the same URL because aliases cannot all be recognized. Already-delivered records skip later downloads and uploads. Manual runs, shared scheduling and mixed local/remote feeds use the same per-destination tracking and History.

A pending record is saved before PUT. If a connection drops after the endpoint stores the file, a later run can recover matching contents by read-back rather than uploading again. Different contents or refused verification stop for review. Failed or unverified publication leaves the remote file untouched and reports that a file may have been stored. Inspect the endpoint before retrying/resetting. Reset includes HTTPS records: existing names receive suffixes, while a destination you manually emptied can reuse original filenames. The reset does not change the endpoint's conditional-create requirements.

First live manual HTTPS checks (reuse existing emails and folders, with schedules and Background Runs OFF):

1. Stop the app, rebuild and reopen it.
2. Test the saved HTTPS location and confirm the account/folder is still accessible through the provider's website. Use the saved upload URL previously tested; a successful connection test alone does not prove uploads.
3. Clear selections, Setup → Outlook Mailbox Accounts to select the existing AttachmentTest source, then Add Saved Location to select HTTPS. Replacement and archiving remain off/disabled.
4. Preview. Confirm Would copy and planned suffixes for occupied names. Refresh the provider's folder and confirm no uploads occurred.
5. Save as Outlook HTTPS Test, then restore it through Saved Feeds → Use Feed if selections clear.
6. Run Now. Confirm Copied results. Download the new remote files through the provider's website, compare contents with the email attachments, and check HTTPS History. Existing files must remain intact.
7. Preview/Run again. Expect Already delivered/Skipped and no additional remote copies.
8. With schedules still off, Reset Attachment Records and run the same emails again. Confirm existing remote copies remain and additional copies receive suffixes.
9. Add the existing local destination, re-save and run. Confirm HTTPS skips completed attachments while the local destination receives matching copies; review separate History results.
10. Restart, restore the feed and Preview. Confirm source/destination choices and progress persist, with emails unchanged. Leave the feed unscheduled for the next test stage.

Outlook metadata access uses its existing read-only mailbox permission. OneDrive source and upload connections are managed separately: sources use Files.Read and upload destinations request Files.ReadWrite. See Upload files to OneDrive destinations for the first upload stage. Outgoing email is on hold; SharePoint libraries are excluded.

What the application does

The main window separates **Source Files After Delivery** from **Existing Destination Files**. Archiving chooses what happens to a successfully delivered source. Replacement chooses what happens to an existing same-name destination file; leaving replacement unchecked skips it even if its contents differ. The explanation below the replacement checkboxes updates with your choices: all unchecked means skip everywhere, selected types are listed for replacement, and remaining types are listed for skipping. These choices still require the destination's replacement support. Hover over each checkbox for its specific explanation. Saving choices in the main window does not update an existing saved feed: use **Save Feed** to do that, then review/re-enable its schedule. After updating a feed with a saved schedule, the main status message reminds you that automatic runs are off.

In Saved Locations, a bold **Saved password: available** (or token/passphrase) message means the hidden value is still stored. You can leave the blank box alone when keeping the same authentication method. Enter a new value only to change it. **Save And Test** uses that stored value after saving. A **No saved...** message means that authentication method needs a value, except an optional private-key passphrase. Connection tests do not upload files or prove write permission.

Copy files from one local, network, saved SFTP, FTP, FTPS or compatible HTTPS/WebDAV source folder to one or more local folders, network folders, SFTP servers, FTP/FTPS servers, or compatible HTTPS endpoints. You can combine destination types in one run. By default, originals stay in the source folder, and delivered files keep their names and contents. For local/network sources, optional **Archive Delivered Files To Used Files** moves successfully delivered source files into a Used Files subfolder after every selected destination succeeds. Remote sources remain on their server; remote source archiving is not available yet.

Only files directly inside the selected source folder are included. Subfolders, including Used Files, are not copied. Saved SFTP, FTP, FTPS and compatible HTTPS/WebDAV locations can be selected with the source's **Use Saved...** button and stored in feeds. Folder links, junctions and file links are not supported. Manual runs start with **Run Now**. Named feeds can also run on interval, daily, weekly or one-time schedules while the app remains open, including hidden in the system tray. Optional **Background Runs** allows enabled scheduled feeds to run with the app closed while you remain signed into Windows and the computer is awake. Synchronization, direct moves without a source archive and additional HTTPS API formats are future work.

Inspect SFTP source files

In **Saved Locations**, select a saved SFTP location, then click **Inspect Source Files...** below the location list. The same button supports saved FTP, FTPS and compatible HTTPS/WebDAV locations. SFTP inspection uses its saved connection settings and hidden credential, including a previously verified SHA256 server fingerprint. Unsaved edits in the window are ignored; save a change before inspecting it.

The SFTP Source Files window shows the saved folder, file names, sizes in bytes and modification times in UTC. You can select and copy table cells with Ctrl+C. Only ordinary files directly inside the saved folder are listed. Subfolders (including Used Files), symbolic links and this application's `.filerouting-` temporary entries are excluded. An empty folder produces a successful empty listing. Folders with more than 10,000 entries must be narrowed before inspection. Close and inspect again to refresh the snapshot.

This inspection window lists files only. It does not download, upload, archive, rename or delete anything, change your main-window source, or save a feed. A successful listing does not prove download permission or verify file contents. To deliver from this saved SFTP folder, select it with **Use Saved...** in the main window's Source folder section as described below.

Inspect HTTPS source files

Select a saved HTTPS location in Saved Locations and click **Inspect Source Files....** Inspection lists a WebDAV folder using its saved URL, authentication method and hidden credential. It does not download file contents, change files, select a main-window source or create a feed. To download and deliver files, select the location through the main source's Use Saved button as described below. Existing HTTPS destination uploads remain available.

The server must support WebDAV PROPFIND with Depth 1. A normal webpage or download URL alone cannot list a source folder. Files.com supports this interface at the site's HTTPS folder URL, such as `https://nmsoftworks.files.com/DeliveryTest/`; the browser navigation URL containing `/files/` is not the WebDAV folder URL. Listing access does not prove download permission. Login requirements, WebDAV access settings and folder permissions must allow the saved authentication method. Connection Test still tests HEAD, whereas Inspect

Source Files tests folder listing.

The HTTPS Source Files window shows top-level ordinary names, available byte sizes and modification times in UTC. Missing sizes/times are blank. Subfolders, unsupported resource types and `.filerouting-` staging entries are excluded. The listing is bounded to 10,000 child entries and 4 MiB of XML; unsafe Windows filenames, duplicate/case-only names, unexpected paths/origins and malformed metadata stop inspection. Redirects are not followed; trusted TLS is required. Close and reopen inspection to refresh. Table cells can be copied with Ctrl+C.

First live Files.com inspection check:

1. Disable Background Runs before rebuilding. Rebuild and open the app.
2. In the Files.com website, upload `https-source-check.txt` containing `First version` directly into DeliveryTest. Put a different disposable file inside a subfolder of DeliveryTest.
3. Open Saved Locations and select the existing HTTPS Files Test location. Confirm its saved HTTPS folder URL is `https://nmsoftworks.files.com/DeliveryTest/` (a trailing slash is optional), with its working authentication. Save any changes before inspecting; the saved password may stay blank when retaining it.
4. Click Inspect Source Files. Confirm the HTTPS Source Files window opens and names the correct saved location/folder.
5. Confirm `https-source-check.txt` appears with the correct byte size if reported. The nested file should not appear. Other ordinary files already in DeliveryTest may also appear.
6. Select the test file row and press Ctrl+C. Confirm its information can be pasted into a text window.
7. Close inspection, upload another new top-level disposable file through Files.com, and inspect again. Confirm the new name appears and both remote originals remain unchanged.
8. Select a saved local folder and confirm Inspect Source Files is disabled. Selecting HTTPS as the main source is a separate action described below.

Deliver from an HTTPS source

Choose **Use Saved...** in the main window's source section, select a saved HTTPS/WebDAV location, and click **Use As Source**. Its name and folder URL appear in the source field. Archiving is turned off and disabled; remote originals remain on the server. Browse to a local source to make archiving available again. HTTPS/WebDAV folder listing and authenticated GET downloads are required; an ordinary website, Rapaport login endpoint, or custom JSON API is not automatically a compatible source folder.

Preview lists ordinary top-level names and checks destination collisions without downloading source file contents. Run lists again and requires every source file to have a reported size and a strong file version identifier (ETag) in WebDAV metadata. It downloads each file to a new temporary local folder with an If-Match condition for that version, requests unchanged bytes, and checks received byte counts. GET size/version headers are checked when supplied; if absent, byte counts and the final WebDAV listing still must verify the original size/version. Changed versions, missing files, incomplete/extra/encoded/partial responses and failed downloads stop the entire preparation before any file is delivered to any destination. Source download failures are not automatically retried.

After all downloads, the app lists again and checks each original file's size, strong version and previously reported modification time. This depends on the endpoint providing truthful metadata and honoring conditional requests; it does not lock the server or prove a producer has finished writing a file. Keep incoming uploads separate and place completed, stable files in the pickup folder. New names arriving after the initial listing are picked up on a later run. All original top-level files in the selected folder are included, and downloads may still occur for names whose destinations subsequently skip.

Normal local/network and remote destination copy, skip and replacement choices apply after preparation. Save Feed, Use Feed, restart, in-app schedules and Background Runs retain the HTTPS source reference and resolve the saved location's current settings. Source failures are recorded in History and disable scheduled feeds for review. History identifies the remote source rather than the temporary directory. No source upload, rename, deletion or archive is performed.

Keep enough local disk space for all source files. Temporary downloads normally clear after use; a crash can leave files under the application's HttpsDownloads data folder. Later runs do not reuse them. Cleanup warnings identify remaining copies for review. HTTPS source and destination URLs cannot identify the same server folder; use separate storage, including when different aliases/protocols could reach the same files. Current selections use schema 9 and feeds use schema 11; earlier builds refuse these newer formats.

First live Files.com source delivery check:

1. Disable Background Runs and other test schedules. Rebuild, open the app and click Clear Selections.
2. In Files.com, upload `https-source-copy.txt` containing First version directly into DeliveryTest. Put another disposable file in a subfolder. Existing ordinary top-level files may also be delivered during this test.
3. Choose the source's Use Saved → HTTPS Files Test → Use As Source. Confirm its name/URL and that archiving is unchecked and disabled.
4. Add an empty local destination, leaving replacement unchecked. Preview should show Would Copy for the test file, exclude the nested file and leave the destination empty.
5. Run Now. Open the local copy and confirm First version. Confirm the remote original remains; History should show Copied and the HTTPS source.
6. Using Files.com, replace the remote `https-source-copy.txt` with one containing Second version under the same name. Wait until its upload completes.
7. Preview and Run should skip the existing destination name. Confirm the local file still contains First version and the remote file contains Second version.
8. Check Replace Existing Folder Files. Preview should show Would Replace; Run should report Replaced and the local file should contain Second version.
9. Add a second empty local destination and run. Confirm Second version in both and the remote original still present.
10. Save as HTTPS Source Test, restore through Saved Feeds → Use Feed, then restart. Confirm the HTTPS source/destinations and folder replacement setting remain, with archiving disabled. Browse to a local source and confirm archiving becomes available. Leave the test schedule disabled. Live scheduling and remote-destination tests follow separately.

Deliver from an SFTP source

Click **Use Saved...** beside Source folder, select a saved SFTP location and click **Use As Source**. The source field displays its saved name, server and folder. The location must have working saved credentials and a verified SHA256 server fingerprint. Unsaved location edits are ignored. Selecting SFTP turns archiving off and disables that checkbox; originals remain on the server. To return to local/network delivery, choose a folder with Browse or Use Saved.

Add one or more separate destinations as usual. The source and destination must point to different storage folders; do not use aliases or another protocol to point a destination back at the source. The app blocks a directly matching SFTP source/destination folder, including basic path normalization and matching saved server fingerprints. It cannot identify every server-side folder alias or mapping across protocols.

Preview lists ordinary top-level source files and checks destination names. It creates temporary local name placeholders but never downloads source contents or publishes files. Download permission is still untested. Subfolders, links and application staging files are excluded. Existing names show Would Skip unless the corresponding destination replacement choice is checked.

Run Now lists the source again and downloads its files to a unique temporary local folder before starting any destination delivery. Each downloaded byte count must match the listing, and file sizes and modification times must still match a second listing. A missing, incomplete or detectably changed file fails preparation without starting destination delivery. SFTP does not lock the server files: changes that retain the same size and timestamp may be undetectable. Arrange for another program to finish writing source files before running the feed.

All destinations receive the downloaded contents using the existing copy, skip and replacement settings. Files already present at a destination still skip by default even if their contents differ. The initial implementation downloads the listed files before these final delivery checks, so a skipped file can still be downloaded temporarily. Remote originals are not archived, renamed or deleted. Names and contents are preserved; destination timestamps and other metadata are not guaranteed to match the remote source.

Temporary downloads are cleaned up after a preview or run. A cleanup warning means files may remain in the displayed temporary folder; review it rather than repeating delivery just for cleanup. An app crash or power loss can also leave temporary downloads under the application's SftpDownloads data folder. These folders are not reused for later runs. Keep enough free local disk space for the entire source listing. Filenames Windows cannot safely represent, including reserved names, alternate-stream names and case-only duplicates, block preparation until renamed.

Save the combination as a feed to retain its SFTP source reference and delivery policies. Restart and Use Feed restore the source type. Manual and scheduled runs resolve the saved location's current settings. Deleting the location or changing its type requires choosing another source. SFTP preparation failures appear as Failed in History; scheduled failures disable the affected feed for review. Destination connection retries retain their existing behavior; source downloads are not automatically retried. History records the SFTP source rather than the temporary local path.

For the first live test, use a small disposable SFTP folder and local destination:

1. Disable Background Runs, rebuild and open the application. Click Clear Selections. Leave other feed schedules disabled for this test.
2. In FileZilla, put `sftp-source-copy.txt` containing First version directly in your test SFTP folder. Put another disposable file in a subfolder.
3. Click the source's Use Saved button, select that saved SFTP location and click Use As Source. Confirm its name/server/folder appear. Archiving should be unchecked and disabled.
4. Add an empty local destination folder. Leave all replacement choices unchecked. Preview should show Would Copy for the top-level file, but exclude the nested file. The local destination should remain empty.
5. Click Run Now. Open the local copy and confirm First version. In FileZilla confirm the original remains unchanged. History should report Copied and show the SFTP source.
6. Change the remote file to Second version under the same name using FileZilla. Preview and Run Now should skip it; the local destination should retain First version.
7. Check Replace Existing Folder Files. Preview should show Would Replace. Run Now should report Replaced and the local file should contain Second version. The remote original should remain.
8. Add a second empty local destination. Run Now should replace at the first destination and copy to the second. Confirm Second version in both and the source still present remotely.

9. Save this as SFTP Source Test. Reopen it using Saved Feeds → Use Feed, and restart the app. Confirm the SFTP source and replacement setting are retained and archiving remains unavailable.
10. Select a local source using Browse. Confirm the archiving checkbox becomes available again. Disable any test schedule before ending this first test. Scheduled and live remote-destination SFTP-source testing follow separately.

To test this stage with disposable files:

1. Disable Background Runs before rebuilding. Rebuild and open the application.
2. In FileZilla, connect to your test SFTP account and put `source-check-a.txt` and `source-check-b.txt` directly in a test folder. Create a subfolder with another file inside it.
3. In Saved Locations, select a saved SFTP location pointing to that test folder. If it is new, save its credentials and verified server fingerprint and complete Test Saved Connection first.
4. Click **Inspect Source Files....** Confirm that the window names the correct saved location and folder.
5. Confirm both top-level files appear with sizes and modification times. The subfolder and its file should not appear. Compare the names and sizes with FileZilla.
6. Select a file row and use Ctrl+C to copy its information. Close the inspection window and confirm the files remain unchanged in FileZilla and the main-window source has not changed.
7. Add another top-level test file using FileZilla. Inspect again and confirm it appears in the refreshed listing.
8. Select a saved local folder. Confirm **Inspect Source Files...** is disabled. HTTPS/WebDAV inspection is described in its own topic. FTP/FTPS inspection is described in its own topic below.

Inspect FTP and FTPS source files

In **Saved Locations**, select a saved FTP or FTPS location and click **Inspect Source Files....** Inspection uses the saved folder (including / for the account-visible root), username, hidden password, port and Passive/Active mode. FTPS also uses the saved Explicit/Implicit mode, requires a trusted certificate and requires an encrypted data connection. Plain FTP is unencrypted. Save edits before inspection; the operation reloads the saved settings.

The inspection window shows ordinary top-level filenames, reported byte sizes and reported modification times. Blank size or date cells mean the server did not supply that information. FTP/FTPS dates are displayed as reported by the listing parser and may not be UTC. Subfolders, recognized links and application `.filerouting-` temporary entries are excluded. An empty folder is a successful empty listing. Folders with more than 10,000 parsed entries must be narrowed. Select rows and press Ctrl+C to copy information. Close and inspect again to refresh it.

Inspection does not download contents, upload, rename, archive or delete anything. The FTP/FTPS upload-permission checkboxes do not need to be enabled to inspect a source. Listing access does not prove download permission. To deliver these files, select the saved FTP/FTPS location with the source's Use Saved button as described in the next topic.

Test FTP and FTPS separately when working accounts are available:

1. Disable Background Runs, rebuild and open Saved Locations.
2. In FileZilla, connect to your FTP test account. Place two disposable text files directly inside a test folder, plus another file inside a subfolder.
3. Select the saved FTP location for that folder in the app. Confirm Test Saved Connection passes first.
4. Click Inspect Source Files. Confirm the correct saved location/folder, top-level filenames and reported sizes. The subfolder's file should not appear. Dates may use the server's time zone.

5. Select a row, press Ctrl+C and paste its information. Close the window and confirm the server files remain unchanged in FileZilla.
6. Add another top-level file using FileZilla, inspect again and confirm the refreshed listing includes it.
7. Repeat the connection and inspection with a saved FTPS location and its correct security mode. Check that the window title identifies FTPS. If no FTPS account is available, leave this check pending.
8. Select a saved local folder and confirm Inspect Source Files is disabled. HTTPS/WebDAV inspection is described in its own topic.

Deliver from an FTP or FTPS source

Use **Use Saved...** beside Source folder to select a saved FTP or FTPS location, then click **Use As Source**. The source field identifies its protocol, name, server and folder. Saved authentication, port, Passive/Active data mode and Explicit/Implicit FTPS mode are reused. FTP is unencrypted; FTPS requires trusted TLS and encrypted data transfers. Source upload-permission checkboxes do not grant or control download permission.

Archiving turns off and is unavailable for remote sources. Originals remain on the server. Select distinct destination storage; do not point a destination back at the source through another protocol or folder alias. Directly matching source/destination paths on the same hostname are blocked, including matching paths across FTP/FTPS ports, but every server-side alias or mapping cannot be recognized. Select a local/network source to make archiving available again.

Preview lists top-level ordinary names and checks destination collisions without downloading file contents. Subfolders, recognized links and application staging entries are excluded. A successful preview does not establish download permission. Run Now lists again and downloads all selected source files into a new temporary local folder before any destination delivery. Transfer uses binary mode to preserve file bytes. The server must supply a usable SIZE response; unknown or refused sizes block delivery, even if names can be listed. Received byte counts must match, sizes are checked again, and reported modification times are compared with a final listing. Incomplete, failed, growing or detectably changed sources stop preparation without delivering earlier downloaded files.

Server files cannot be locked by this application. Same-size changes with an unchanged or unavailable timestamp may escape detection. Have another program finish writing incoming files before running a feed. Remote originals are retained; temporary copies are cleaned up after use. Keep enough local disk space for all top-level source files. Temporary folders can remain after a crash under the application's FtpDownloads data folder; later runs do not reuse them. Cleanup warnings identify leftover files for review. Destination metadata is not guaranteed to match the server.

Local/network and all existing remote destination types use their normal copy, skip and replacement policies after preparation. Downloads may occur even when destination names subsequently skip. Save Feed, Use Feed, restart, in-app schedules and Background Runs retain the source protocol/reference and resolve the saved location's current settings. A deleted source location or changed protocol requires selecting the source again. Preparation failures produce Failed History and disable scheduled feeds for review. Source downloads are not automatically retried; existing destination-check retries remain unchanged. History shows the remote source rather than the temporary folder.

Use disposable files to verify FTP first, then FTPS:

1. Disable Background Runs and other test schedules, rebuild and open the app. Click Clear Selections.
2. In FileZilla, upload ftp-source-copy.txt containing First version directly into your FTP test source folder. Put another file in a subfolder.
3. In the main source section, choose Use Saved → the saved FTP location → Use As Source. Confirm the source identity and that archiving is unchecked/disabled.

4. Add an empty local destination. Leave replacement unchecked. Preview should show Would Copy for the top-level file, exclude the nested file and leave the destination empty.
5. Run Now. Open the local copy and confirm First version. Confirm the remote original remains and History reports Copied with the FTP source.
6. Use FileZilla to replace the source contents with Second version under the same name. Preview and Run should skip; the local destination should retain First version.
7. Check Replace Existing Folder Files. Preview should show Would Replace; Run should report Replaced and the local file should contain Second version. Confirm the remote source remains.
8. Add a second empty local destination and run. Confirm Second version at both destinations and the original retained remotely.
9. Save as FTP Source Test. Load it with Use Feed and restart. Confirm source protocol and replacement policy are retained. Browse to a local source and confirm archiving becomes available again.
10. Repeat steps 2-9 with a saved FTPS connection and a fresh `ftps-source-copy.txt` file. Use that location's configured security mode and port, and save as FTPS Source Test. If an account is unavailable, report that protocol test as pending. Scheduled and remote-destination verification follows separately.

By default, existing destination names are skipped, even if source contents have changed. Separate **Replace Existing Folder Files**, **Replace Existing SFTP Files**, **Replace Existing FTP/FTPS Files**, and **Replace Existing HTTPS Files** checkboxes enable replacement for their destination types. Remote replacement requires server support. Protection depends on the settings and server behavior described below.

Unless a procedure is specifically testing archiving, leave **Archive Delivered Files To Used Files** unchecked when verifying that files remain directly in the source folder.

Start the application

For the current development build:

1. Open `AutomatedFileRouting.slnx` in Visual Studio with .NET 10 desktop development support installed.
2. Set `FileRouting.Desktop` as the startup project.
3. Build the solution.
4. Start the application.

Keep running in the system tray

Click **Hide To Tray** to hide the main window while keeping the app running. Enabled feeds continue on their schedules while the computer is awake. Hiding does not enable any disabled schedules or change current selections. The button is unavailable during another operation; close any configuration windows first.

Find the application icon near the Windows clock; it may be under the hidden-icons arrow. Double-click it, or right-click and choose **Open Application**, to restore the main window. Reopen the existing app this way rather than starting another instance. Windows may suppress the informational tray notification; the icon still provides access.

To stop the app, right-click the tray icon and choose **Exit Application**, or close the main window with its X. Closing with X exits rather than hiding. Exit restores the window first and waits for any current operation to finish; retry Exit afterward. A successful exit removes the tray icon and stops the in-app scheduler. If **Background Runs** is enabled, scheduled feeds

can continue after exit. Disable Background Runs first if you want closing the app to stop automatic deliveries. The main window does not start automatically with Windows. No scheduled delivery can run while the computer is asleep.

Run schedules with the application closed

Background Runs is optional and starts OFF. Open it from the main window to see its status, enable or update it, disable it, and read the last background result. Enabling it applies to **all saved feeds with enabled schedules**. Review Saved Feeds first. It does not enable a disabled schedule or change delivery choices.

A feed's schedule and Background Runs are two separate settings:

- **Enable automatic runs** in a feed's Schedule window turns on that feed's schedule. With Background Runs OFF, it runs while the application is open or hidden in the tray. The Saved Feeds window does not need to stay open; close it after saving the schedule so the in-app scheduler can continue.
- **Enable / Update Background Runs** also allows those enabled feeds to run when the application is fully closed. It does not turn on disabled feed schedules. You must remain signed into Windows, and the computer must be on and awake.
- To stop one feed in both modes, uncheck its **Enable automatic runs** setting and save. To stop only closed-app scheduling for all feeds, choose **Disable Background Runs**; enabled feeds can still run while the application is open or hidden in the tray.

You must remain signed into the same Windows account and keep the computer on and awake. Locking the screen is allowed; signing out is not. No Windows password is requested or saved. The Store version uses a package-managed background helper with normal user permissions. The unpackaged development version uses Windows Task Scheduler's [signed-in interactive mode](#). Neither mode wakes the computer or installs a Windows service.

Windows checks for due feeds about once a minute. When the app is open or hidden in the tray, the in-app scheduler takes over. A shared lock allows only one scheduler to deliver files at a time. If you open the app during a background delivery, its controls briefly wait until that run finishes. While the app owns scheduling, its existing busy-window rules still apply.

Both modes use the same feed schedules, saved credentials, replacement and archiving choices, connection retries and History. Background deliveries appear as **Scheduled** in History. Overdue schedules attempt one run rather than replaying every missed interval. One Time schedules are consumed before delivery. Delivery, archive or history-saving failures disable the affected feed schedule for review. An interrupted run is also disabled when scheduling next starts; inspect the destinations before re-enabling it.

For network sources and destinations, prefer a full network path such as \\ComputerName\ShareName\Folder instead of a mapped drive such as Z:. A drive available in your main window may not be available to the background process. The same Windows account must have access to the share. An unavailable source or destination can fail the run and disable its schedule.

Disable Background Runs stops future closed-app checks. A run already in progress may finish. Enabled feed schedules still run while the application is open. To stop a particular feed in both modes, disable its schedule in Saved Feeds.

In the **Microsoft Store version**, enabling Background Runs starts a small helper and permits Windows to start it when you sign in. It continues checking while the main window is closed, and hands delivery scheduling to the visible application while it is open. The helper uses the same agreement, feed claims and failure checks. It does not open a sign-in browser or accept the agreement. Missing consent, a disabled Windows startup entry or an invalid agreement stops new background checks. A failure that prevents safely saving schedule state stops the helper for review; use Setup → Background Runs to restart it after correcting the problem.

You can also disable **Automated File Routing Background Runs** in **Windows Settings** → **Apps** → **Startup** or Task Manager. The app respects that choice; turn the Windows setting back on before enabling Background Runs again. The helper can take up to about a minute to notice a changed setting; a current delivery can finish. Refresh Status reports whether the helper is running. No separate Windows Task Scheduler job is created by the Store version. Windows manages the startup registration with the installed package, so it does not retain an executable path into an older package version.

Before a **Store update**, disable Background Runs, wait for its helper to stop, exit the app, then update through Microsoft Store. Reopen the updated app, review your feeds and enable Background Runs again. Enabled feed schedules are retained separately. After an update or sign-in, the helper runs only if both its saved opt-in and Windows startup permission still allow it. Uninstalling the Store package removes its package startup registration; other installations' tasks are left alone. Uninstalling or resetting an app can remove its local settings, so keep any needed records before doing so.

For the **unpackaged development version**, the Windows task points to the current application files. Disable background runs before rebuilding, moving or replacing those files, and before restoring an older checkpoint. After rebuilding or moving the application, open it and choose **Enable / Update Background Runs** to use that build. Keep the complete application folder available. If Windows policy prevents setup, the window reports the error; leave background runs off and use the in-app scheduler.

For the first test, use disposable local folders:

1. Rebuild and open the application. In Saved Feeds, disable other enabled schedules before this test because background runs apply to all enabled schedules.
2. Choose a new empty source folder and a different empty local destination folder. Check **Archive Delivered Files To Used Files** and leave all replacement choices unchecked. Save a feed named **Background Test**.
3. In Saved Feeds, select Background Test and configure an enabled Custom interval of one minute. Save the schedule and close Saved Feeds.
4. Open **Background Runs**. Confirm it initially says OFF. Click **Enable / Update Background Runs**, confirm the message, and check that status becomes ON. Close that window.
5. Close the application's main window with its X. Stay signed into Windows and keep the computer awake.
6. After closing the app, create `background-check.txt` directly in the source folder with contents `First version`.
7. Wait about two minutes. Check that the destination contains the file with First version. Check that the source file is now in the source folder's Used Files subfolder.
8. Reopen the app. In History, confirm a Scheduled delivery with Copied and an Archived source result. In Saved Feeds, check its last result and next run. In Background Runs, check the last background result.
9. Click **Disable Background Runs** and confirm the status becomes OFF.
10. Disable the Background Test feed's schedule in Saved Feeds before ending the test. Re-enable other schedules only when you are ready.

To verify tray operation:

1. Add a fresh test file to a tested local feed's source folder.
2. In **Saved Feeds**, enable that feed's **Custom** interval at one minute and save.
3. Close Saved Feeds, then click **Hide To Tray**.
4. Leave the computer awake for approximately 70 seconds and check that the destination contains the file.

5. Double-click the tray icon. Confirm the main selections are retained and History shows the Scheduled copy.
6. Hide again, then right-click the icon and choose **Open Application**. Confirm it restores the same window.
7. Disable the test schedule and save, then close Saved Feeds.
8. Hide again and choose **Exit Application** from the tray menu. Confirm the app closes and its icon disappears.
9. Start the app again and confirm saved selections/history remain and the test schedule remains disabled.
10. Close the main window with X and confirm the tray icon disappears.

First local-folder delivery

1. Create separate source and destination folders. Neither may be inside the other.
2. Put a disposable test file in the source folder.
3. Under **Source folder**, click **Browse...** and choose the source.
4. Under **Destination locations**, click **Add Folder...** and choose the destination.
5. Check the displayed source and destination paths.
6. Click **Preview** and review every row. Preview does not copy or upload files.
7. Click **Run Now** to perform delivery.
8. Review **Delivery Results** and check the destination file.
9. Open **Feeds** → **Delivery History** to inspect the saved run.

Folders must already exist and be accessible. Source and local destination folders must be separate and non-nested; local destinations must also be separate and non-nested relative to each other. Network paths use your current Windows access permissions.

Send the same files to several destinations

One feed can send each source file to several locations in one run. For example, the same file can go to a local folder, an FTP server and an HTTPS upload folder. You can combine local/network folders, SFTP, FTP, FTPS and HTTPS destinations. A separate feed for each destination is not required.

1. Choose one source folder.
2. Use **Add Folder...** or **Add Saved Location** to add a destination.
3. In **Add Saved Location**, check one or more saved locations and click **Add Selected Locations**. You can mix destination types in the same selection. Repeat **Add Folder...** as needed for unsaved folders until every intended destination appears in the main window's **Destination locations** list.
4. Choose replacement options for the destination types you want to replace. Unchecked types skip existing names.
5. Click **Preview** and review the result for each file at each destination.
6. Click **Run Now**, or save this combination as a feed and schedule it. Each run delivers the same source files to all selected destinations.

Results and History report each destination separately. A failure at one destination does not undo a successful delivery elsewhere. A saved location is available for selection; it is not included in a run until you add it to the destination list.

With **Archive Delivered Files To Used Files** checked, each original moves into **Source\Used Files** only after every selected destination reports **Copied** or **Replaced** for that file in the same run. Any **Skipped** or **Failed** result keeps that original directly in Source for review. With archiving unchecked, originals stay in Source. Used Files is excluded from subsequent uploads.

Choose the right source and destinations

The main window's selected destinations determine where files go. Creating a saved location does not automatically add it to that list.

1. Choose a local/network source with **Browse...**, or click **Use Saved...** to select a saved Folder, SFTP, FTP, FTPS or compatible HTTPS/WebDAV profile. The same picker includes configured Outlook mailbox and OneDrive source folders. Remote and Microsoft sources retain their originals.
2. Add an ordinary folder with **Add Folder...**, or click **Add Saved Location** to open the saved destination checklist.
3. Check the box beside each destination you want, reviewing both **Name** and **Path / server**, then click **Add Selected Locations**. The selected count updates as you check/uncheck boxes. Clicking a row only shows its path; use its checkbox to include it.
4. Locations already in the main destination list show **Already Added** and cannot be checked again. Cancel or close the picker to leave the main list unchanged. Repeat to add more destinations. **Use Saved...** for the source remains a single saved-folder selection.
5. To remove a destination from this run configuration, select it and click **Remove Selected**. This does not delete its saved profile or destination files.
6. Before Preview or Run Now, confirm the selected protocol, server and path for every destination.

To start a different combination, click **Clear Selections** on the main window. It clears the source and all destinations, including their saved-location references, and saves the empty selection for reopening. It does not delete saved feeds, schedules, locations, credentials, files or history. If saving fails, selections remain and an error is displayed. If settings could not be loaded, clearing applies only to the current session.

Selections, saved profiles and history persist after reopening the application. When a saved profile is selected, its current saved settings are used for delivery. If a referenced profile was deleted, correct the selection before running.

Replace existing folder files

The main window's **Replace Existing Folder Files** checkbox is unchecked by default. Check it to replace same-name files at local/network folder destinations with the source contents. Existing files are replaced even when their contents already match; this is not a changed-files-only synchronization mode. New destination names are copied normally. Source files remain unchanged.

This option affects only folder destinations. SFTP, FTP/FTPS and HTTPS have separate replacement checkboxes. Source handling is separately controlled by Archive Delivered Files To Used Files.

Preview shows **Would replace** for an existing ordinary folder file. Directories and file links are never replaced. **Run Now**, scheduled results and History show **Replaced**, separate from Copied. Preview makes no changes, and the destination is checked again during delivery.

Replacement stages the full source in a temporary file in the destination folder, verifies its size, then asks the filesystem to replace the target. There is no delete-then-copy fallback. If the filesystem or network server cannot perform replacement, or the destination is locked, delivery fails. Connection loss during publication can leave the result uncertain; inspect the destination before retrying. Previous destination contents are not backed up and cannot be recovered from History. Use disposable files when testing.

Changing the checkbox saves the current selection setting. **Save Feed** stores it with that feed; changing the main checkbox does not alter a previously saved feed. **Use Feed** restores the feed's choice, which also applies to its scheduled runs. Replacing a saved feed still disables its schedule for review. **Clear Selections** and default clear-after-save reset the checkbox to unchecked. A setting-save failure restores the previous checkbox choice.

To verify replacement:

1. Use disposable source/destination folders. Put `replace-test.txt` containing `First version` in the source.
2. Select those folders, leave **Replace Existing Folder Files** unchecked, then Preview and Run Now. Expect `Would copy` and `Copied`.
3. Edit only the source file to `Second version`, retaining its name.
4. Preview and Run Now with the option still unchecked. Expect `Would skip` and `Skipped`; the destination must still contain `First version`.
5. Check **Replace Existing Folder Files**, then Preview. Expect `Would replace`. Confirm the destination still contains `First version` before running.
6. Run Now. Expect `Replaced`. Confirm source and destination both contain `Second version`; History should show `Replaced` in its count and delivery details.
7. Save a disposable feed with replacement selected. Default clearing should reset the main checkbox. Use **Use Feed** to restore it and confirm the checkbox is checked.
8. Restart and confirm the restored checkbox and saved feed policy remain selected. **Clear Selections** and confirm the checkbox resets to unchecked.
9. Use the disposable feed again, change source contents to `Third version`, and enable a Custom one-minute schedule. Close Saved Feeds and wait about 70 seconds.
10. Confirm destination contents changed to `Third version` and History shows `Scheduled/Replaced`. Disable the test schedule and save afterward.

Current selections write schema 6 and read 1-6; feeds write schema 8 and read 1-8. Older feeds do not enable replacement or archiving automatically. Archive-enabled runs write History schema 2; ordinary runs write schema 1, and this build reads both. Older application builds refuse the new settings/feeds/archive history; restoring an older code checkpoint may require compatible saved data as well.

Replace existing SFTP files

Replace Existing SFTP Files is separate from folder and FTP/FTPS replacement and unchecked by default. It applies only to SFTP destinations and persists in current selections and saved feeds. **Use Feed** restores all replacement choices; **Clear Selections** and default clear-after-save reset all. Scheduled feeds use their saved choices, not the current main-window checkboxes. **Save Feed** again to change a previously saved feed; replacing a feed disables its schedule for review.

Existing ordinary SFTP files preview as **Would replace** when selected; preview does not upload, change or remove them. Directories, symbolic links and other non-regular entries are skipped. New filenames still use the ordinary no-overwrite

upload. With a selected replacement, even identical contents are uploaded again.

The app stages a complete upload under a unique temporary name, verifies its size, rechecks the existing entry, then requests the server's POSIX rename extension to publish the replacement. [SSH.NET's implementation](#) provides this separate operation. The old target is never deleted first and there is no delete-then-upload fallback. Servers lacking the extension or permission report Failed; staging cleanup is attempted. Preview and connection testing do not prove extension support or replacement permission.

Success records **Replaced** in results and History and retains the source. No backup of previous destination contents is made. Replacement publishes the staged file's server-assigned attributes; do not assume it preserves the old file's permissions or timestamps. Avoid other programs changing the same target during the run; the type checks and publication are separate operations. A lost publication response reports uncertainty and requires inspection, with no automatic upload retry. Cleanup failures name the temporary file that may remain.

To verify against a disposable file on your tested SFTP server:

1. Create a fresh source filename such as `sftp-replace-test.txt` containing `First version`.
2. Choose its source folder and only the saved SFTP destination. Leave all replacement checkboxes unchecked.
3. Preview and Run Now: expect `Would copy/Copied`. Download the remote file with FileZilla to a separate verification folder and confirm `First version`.
4. Change only the original source contents to `Second version`, retaining its name. Preview and Run Now with SFTP replacement unchecked: expect `Would skip/Skipped`.
5. Check **Replace Existing SFTP Files**, then Preview: expect `Would replace`. The remote file should still contain `First version` before running.
6. Run Now: expect `Replaced`. Download a fresh verification copy, replacing the earlier verification download only, and confirm `Second version`. Confirm the original source still exists with `Second version` and History records `Replaced`.
7. If replacement reports Failed, inspect its details and remote file. Report the message; leave replacement off until the server's support/permissions are understood. Do not manually delete the target to make this test pass.
8. Save a disposable test feed with SFTP replacement selected. Default clearing resets all replacement choices; Use Feed restores SFTP replacement while folder replacement remains unchecked.
9. Restart and confirm the choices and saved feed policy remain; Clear Selections should reset all replacement choices.
10. Use the test feed again, change source contents to `Third version`, and enable Custom one-minute scheduling. Close Saved Feeds and wait approximately 70 seconds.
11. Download a fresh verification copy and confirm `Third version`; History should show `Scheduled/Replaced`. Disable the schedule afterward and leave the SFTP replacement choice off for ordinary skip tests.

Archive delivered source files to Used Files

Archive Delivered Files To Used Files is optional and unchecked by default. It applies to manual and scheduled runs, including mixed Folder/SFTP/FTP/FTPS/HTTPS destinations. Each source file must report **Copied** or **Replaced** at every selected destination in the same run before it can be archived. If any destination skips or fails, that file stays directly in the source folder. Other files whose destinations all succeed can still be archived.

For example, with source `C:\Delivery\Source`, the archive is `C:\Delivery\Source\Used Files`. The application creates it when needed. Only files directly inside Source are scanned; Used Files and every other subfolder are excluded. Keep selecting Source for future deliveries. Selecting Used Files itself as a source would intentionally expose those

archived files for delivery.

The archive normally retains the filename. If that name already exists in Used Files, the new archive gets a unique suffix; existing archived files are never replaced. Results and History give the actual archive path. Archive file attributes, permissions and timestamps may differ from the source; contents are verified. Used Files is on the same storage as the source and is not an off-device backup. Manage its space and retention yourself; the app does not automatically purge it.

The application holds a read lock while delivering each archive-enabled source to its destinations. Finish writing/close producer files before running. It verifies a recoverable archive copy against the delivered source contents before removing the original from the top-level source folder. An active writer, changed source, source/archive link, missing permissions, insufficient space, a removal lock or unsupported filesystem operation leaves the source in place and reports an error. Network source folders require environment-specific testing.

Delivery results stay Copied/Replaced if only archiving fails. A separate **Archive Failed** count and source-handling details explain that failure; a published archive copy may also exist. Inspect both copies before retrying. Scheduled archive failures disable the schedule for review. If a run is interrupted, some completed files may already be in Used Files while others remain in Source; inspect destinations and Used Files before resuming. History failure does not undo deliveries or archives. No repeat delivery is needed just to create a History record.

A successful earlier run is not enough to archive a later skipped file. After a partial multi-destination delivery, retrying with skip behavior may leave the source because earlier successful destinations now skip. Review the destinations and replacement choices before attempting another run; there is no automatic inference that existing destination contents match.

The choice saves with current selections and named feeds, restores with Use Feed, and persists after restart. Clear Selections/default clear-after-save turn it off. A scheduled feed uses its saved choice. Save Feed again to change it, then review/re-enable the disabled schedule in Saved Feeds → Schedule. Saving a new feed does not change another feed's schedule. Preview explains planned archiving but creates no archive folder and changes no files.

Verify local archive behavior

Use disposable files and separate source/destination folders.

1. Rebuild/start. Create a fresh Source folder with only `archive-check.txt` containing `First version`, plus two separate empty destination folders A and B.
2. Clear Selections, choose Source, add A and B. Leave archiving and replacement unchecked. Preview/run: both destinations copy and the original remains in Source.
3. Check Archive Delivered Files To Used Files and run again with replacement unchecked. Both destinations skip; the original stays in Source.
4. Check Replace Existing Folder Files. Preview reports replacement plus conditional archiving plans, without changing anything. Run Now replaces both destinations, then archives the source.
5. Open Source\Used Files and confirm `archive-check.txt` contains `First version`. It should no longer be directly in Source. Check History's Archived count and source-handling path.
6. Preview again. Used Files contents must not appear in the file list.
7. Create another `archive-check.txt` directly in Source containing `Second version`. Keep archiving and folder replacement checked; run again. Destinations should contain `Second version`, while Used Files retains both `First version` and `Second version` under different archive names.
8. Create that source filename again containing `Third version`. Uncheck folder replacement and add a third empty destination C. Run: A/B skip, C copies, and the `Third version` source remains because every destination did not

copy/replace it.

9. Recheck folder replacement and run again. All destinations replace with Third version and the source is archived. Confirm all earlier archive versions remain.
10. Save a test feed with archiving checked and default clearing. Confirm the checkbox clears, then Use Feed restores it; restart and confirm it stays selected.
11. Put a new filename directly in Source, enable a short interval schedule for that feed, close the dialog and leave the app open. Confirm all destinations receive it, the source is archived and History shows Scheduled plus Archived. Disable the test schedule afterward.

Repeat first-upload/archive verification with each remote destination you intend to use. A remote failure or skipped name must leave its source file in place.

Named feeds (manual runs)

A feed saves one source and its selected destinations as a named combination. It does not save file contents or duplicate credentials. Feeds run manually unless you explicitly enable a schedule. Scheduling requires the app to stay open or Background Runs to be enabled while you remain signed into Windows with the computer awake.

Save and use a feed

1. On the main window, choose the source and add all desired destinations.
2. Check their displayed paths, protocols and servers.
3. Click **Save Feed....**
4. Open the name dropdown to select an existing feed to replace, or type a new descriptive name to create another feed. When a feed has been loaded using Use Feed, its name is filled in automatically. **Clear selections after saving** is checked by default: a successful save clears the main-window source, destinations and loaded-feed name so you can create another feed. Uncheck it to keep them for an immediate Preview or Run Now. Click **Save Feed**. Replacing an existing name requires confirmation and disables its schedule for review. No files are transferred.

The top of the main window displays **Loaded Feed: [name]** after Use Feed or saving without clearing. This identifies the feed being edited; changing selections does not update the saved feed until you save it. Clear Selections resets the display to **Loaded Feed: None**. The loaded-feed name is session context; after restarting, use Saved Feeds → Use Feed to identify the feed again.

5. To restore it later, click **Saved Feeds...**, select its name, and review the source and destination summary.
6. Click **Use Feed**. This replaces the current source/destination selections but does not start a transfer.
7. Check the main window's restored selections, click **Preview**, then click **Run Now** when ready.
8. Review the normal delivery results and History.

Saving under an existing name asks whether to replace that feed. Choose **No** to keep it, or **Yes** to update its source/destination combination. Replacing a feed disables its existing schedule for review; re-enable it only after checking the new source and destinations. Names are unique without regard to capitalization. Editing the main window selections does not automatically update a saved feed; save under its name again to update it.

After resaving/replacing an existing feed, automatic runs stay disabled until you re-enable its schedule:

The Save Feed window and replacement confirmation display this warning and explain where to re-enable the schedule. Saving under a new name does not disable another feed's schedule.

1. Open **Saved Feeds...** and select the feed you just saved.
2. Click **Schedule...** and review its timing and settings.
3. Check **Enable automatic runs**, then save the schedule. For a One Time schedule, choose a new future date/time if needed.

The previous schedule settings are retained for review, but saving the feed does not automatically resume scheduled runs.

Feeds reference saved locations by ID and use their latest settings when loaded and run. The feed summary shows the selections stored when the feed was saved; the main window refreshes referenced location labels when you use it. If a referenced location was deleted, loading fails instead of falling back to the old connection. Re-create the feed with available locations. Actual folder availability is checked when saving, previewing or running; loading alone does not verify access or write permission.

Clearing selections does not delete the saved feed, its schedule, saved locations, credentials or history. The empty current selection is saved for the next restart. Cancelling or a failed feed save does not clear selections. If saving the empty selection fails after the feed was saved, the app retains selections and reports that separate failure.

Feeds remain after restarting the application. They are independent of the last source/destination selection, and loading a feed becomes your current selection. The History screen records the actual paths and delivery results. Scheduled runs also identify their feed and show Scheduled as their trigger; ordinary manual runs show Manual.

Delete a feed

1. Click **Saved Feeds...** and select the feed.
2. Click **Delete Feed**.
3. Confirm deletion, or cancel to retain it.

Deleting a feed removes only that named combination. Source/destination files, saved locations, credentials, current selections and delivery history remain.

Saved Locations: save, edit, test and delete

Open **Saved Locations...** on the main window. Click **New** for a new profile, or select an existing profile to edit it. Each location needs a unique name.

Button	What happens
Save Location	Validates and saves the entry, then clears the fields in this window for a new entry. Does not test.
Save And Test	Saves a remote location, clears the fields in this window, then automatically tests its saved settings and credential. No reselection is needed.
Test Saved Connection	Tests the selected profile's saved values. Unsaved changes in this window are not used.
New	Starts a new entry.
Delete	Asks for confirmation, then deletes the selected profile and its unused stored credential. Delivered files remain.

Folder profiles use **Save Location**; they do not have a remote connection test. Remote connection tests do not upload files and do not prove write permission.

If validation or saving fails, entries remain in this window and no automatic test starts. If saving succeeds but testing fails, the location remains saved. Select it again to correct its settings. The test result identifies the saved location by name.

Why the password field is blank

Passwords, private-key passphrases and bearer tokens are stored in Windows Credential Manager for your Windows user. They are not displayed when a profile is reopened. A blank field does not mean a saved credential was erased.

1. Read the message below the password, passphrase or token field.
2. Leave the field blank to retain a saved credential when the connection type and authentication method are unchanged.
3. Enter a value only when replacing it or creating an authenticated profile.
4. Click **Save And Test** to use the saved credential immediately.

Changing the connection type or authentication method may require a new credential. A private key without encryption needs no passphrase. **Remove saved key passphrase** explicitly removes a stored SFTP key passphrase; do not also enter a replacement in the same edit. A saved credential reference does not prove the credential is still present or valid-testing checks that.

Save a folder location

1. Open **Saved Locations...** and click **New**.
2. Enter a descriptive location name.
3. Choose connection type **Folder**.
4. Enter a full local or network path, or use **Browse....**
5. Click **Save Location**.
6. Return to the main window and choose it through **Use Saved...** for a source or **Add Saved Location** for a destination.

Save an SFTP location

SFTP uses SSH. It is a different protocol from FTP and FTPS.

1. Open **Saved Locations...** → **New** and enter a name.
2. Choose **Sftp**.
3. Enter the remote folder. Use / for the account-visible root, or a path such as /upload. The folder must exist.
4. Enter the server hostname or IP address without a URL prefix or username.
5. Enter the server port, normally 22, and username. Use your provider's values if different.
6. Choose **Password** or **PrivateKey** authentication.
7. For Password, enter the password. For PrivateKey, browse to the key file and enter its passphrase if it has one.
8. Enter the verified SHA256 server fingerprint if already known. Otherwise leave it blank for the first inspection.
9. Click **Save And Test**.

Verify the server fingerprint

A first test with no trusted fingerprint is expected to refuse the connection and display the observed fingerprint. It does not trust the key automatically.

1. Obtain the expected SHA256 fingerprint independently from the server provider, its dashboard, or the person operating the server.
2. Compare it exactly with the observed fingerprint. Do not trust it solely because the application displayed it.
3. Select the saved location again and enter the verified fingerprint in **Server host-key fingerprint**.
4. Leave the saved password/passphrase field blank.
5. Click **Save And Test** again.

If the saved fingerprint later differs from the server key, the app refuses the connection. Verify the reason for a key change before updating it.

A successful SFTP connection test confirms authentication, the trusted server identity and the remote folder. Actual delivery is needed to test writing. Uploads use a temporary name, verify size and publish under the source filename with a non-overwriting SFTP rename.

Save an FTP or FTPS location

FTP sends credentials and files without encryption. FTPS adds TLS encryption and requires a valid, trusted certificate for the hostname. FTPS is distinct from SFTP.

1. Open **Saved Locations...** → **New** and enter a name.
2. Choose **Ftp** or **Ftps** to match the server.
3. Enter the existing remote folder. / is allowed for the account-visible root; a folder name is not mandatory.
4. Enter the server hostname/IP, username and password.
5. For FTPS, choose **Explicit** or **Implicit** security according to the provider.
6. Check the port. Typical values are FTP/explicit FTPS 21 and implicit FTPS 990; use the provider's actual port.
7. Select **Passive** or **Active** data connection mode to match the server/network. Passive is commonly used; changing the mode can affect listing and transfer connectivity.

8. Choose the upload protection setting described below.

9. Click **Save And Test**.

10. Return to the main window and use **Add Saved Location** to select the location for delivery.

FTPS does not downgrade to plaintext when TLS fails. Its data connection is encrypted. A successful FTP/FTPS test checks authentication, access to the requested folder and a data connection for listing. It does not upload a file, prove write permission, or verify the server's overwrite policy.

FTP/FTPS upload protection settings

Allow uploads with filename checks enables delivery without requiring a server administrator. With **Replace Existing FTP/FTPS Files** unchecked, the application checks for an existing filename, skips it if present, uploads under a unique temporary name, checks the uploaded size, checks the final name again, and renames the temporary file.

There is a remaining risk: if another program creates the same final filename between the last check and rename, the FTP server may replace that file. Filename checks alone cannot guarantee prevention. Enable this option only if you accept that risk, and avoid simultaneous uploads of the same name from other programs. FTPS encryption does not remove this limitation.

Administrator confirmed... is an optional stronger setting. Select it only when the server operator/provider confirms that this account rejects overwriting files during **both uploads and renames**. FileZilla inspection and a successful connection test do not establish that policy. Do not select it merely to enable uploads.

If both settings are unchecked, FTP/FTPS uploads are blocked. Either setting permits delivery. Both settings are saved separately for each profile; the practical option does not assert that the server rejects overwrites.

Replace existing FTP/FTPS files

Allow uploads with filename checks in Saved Locations permits uploads. **Replace Existing FTP/FTPS Files** in the main window separately permits replacing existing same-name FTP/FTPS files. It is unchecked by default. When unchecked, existing names still skip even if uploads are allowed.

Replacement requires the upload option enabled and **Administrator confirmed...** unchecked. An actual server policy that rejects overwrites conflicts with replacement; keep a valid confirmation and use another account instead. Do not uncheck a true confirmation merely to bypass the message.

With replacement selected, Preview reports **Would replace** for an existing ordinary file. Preview does not upload or prove that the server accepts replacement. Run Now uploads to a unique temporary name, verifies its size, then asks the server to rename it over the existing file. It never deletes the old target first or falls back to delete-then-upload. A server may refuse this rename; report that failure and inspect the destination. New names are copied normally. Directories, links and unknown entry types are not replaced.

Successful replacement reports **Replaced**, verifies the published size, retains the source, and creates no backup. The resulting server file may have different attributes or permissions. Avoid other programs writing the same filename during delivery. This method depends on the server's rename behavior and does not guarantee atomic replacement across every FTP server. If publication is uncertain, inspect/download the destination before retrying; uploads are not automatically retried.

The choice persists with current selections and saved feeds. Use Feed restores it; Clear Selections and default clear-after-save reset it. Schedules use the saved feed choice. Save Feed again to update an existing feed and review/re-enable its schedule.

Verify FTP/FTPS replacement

Use a disposable file on your working external server. Close saved-location/feed dialogs before running. Refresh FileZilla and download a fresh copy after each run.

1. Rebuild and start the app. In Saved Locations, select the test profile: **Allow uploads with filename checks** checked, **Administrator confirmed...** unchecked unless actually confirmed. Save changes.
2. Create ftp-replace-check.txt containing First version in a source folder containing only test files.
3. Click **Clear Selections**, choose that source, and add only the saved FTP/FTPS destination. Leave all replacement checkboxes unchecked.
4. Preview should show **Would copy** for the fresh name. Run Now should show **Copied**. Download the file and confirm First version.
5. Change the source contents to Second version. Preview and Run Now should show **Would skip / Skipped**. Download again and confirm the destination remains First version.
6. Check **Replace Existing FTP/FTPS Files**. Preview should show **Would replace**, while the destination still contains First version.
7. Run Now. If the server permits replacement, expect **Replaced**. Download again: expect Second version. Confirm the source still contains Second version.
8. Open History and check the Replaced result/count. If step 7 failed, use **Copy Run Details** and share the message; inspect the remote contents before another attempt.
9. Save a test feed with clear-after-save checked. Confirm selections and replacement choices clear. Use Feed to reload it and confirm the FTP/FTPS replacement choice returns. Restart and confirm current selections retain it.
10. Change the source to Third version, enable a short interval schedule for this test feed, close the dialog, and leave the app open. After the scheduled run, download and verify Third version and a Scheduled/Replaced History entry. Disable the test schedule afterward.

Plain FTP verification does not also verify FTPS encryption/certificate behavior; repeat with a working FTPS profile when available.

Save an HTTPS upload location

The current HTTPS method is **HTTP PUT**: each file is sent to its own URL. HTTPS alone does not define an upload method; the endpoint must support this particular contract.

1. Obtain the HTTPS folder/base URL from the endpoint provider, for example `https://example.com/uploads/`.
2. Confirm it supports HEAD requests, PUT uploads, and the conditional header `If-None-Match: *`.
3. Open **Saved Locations...** → **New**, enter a name, and choose **Https**.
4. Enter the HTTPS folder URL. The app appends the encoded filename. Do not include a username/password in the URL, a query string or a fragment.
5. Choose authentication **None**, **Basic**, or **Bearer**.
6. For Basic, enter username and password. For Bearer, enter the token. None requires no credential.
7. Click **Save And Test**.

8. Return to the main window and use **Add Saved Location** to select this HTTPS destination.

A successful connection test means the base URL responded to HEAD using the saved authentication. It does not prove PUT permission or conditional protection. Preview also uses HEAD: an existing file is skipped, and an absent file may be uploaded using conditional PUT. HTTP 412 during upload is recorded as skipped.

The endpoint must honor If-None-Match: *; a server ignoring it may overwrite files. After upload, the app uses HEAD to compare the remote Content-Length with the source size. Size verification alone is not a comparison of file contents. If HEAD cannot verify the size, the app uses an authenticated GET to read the uploaded file back and compare every byte with the source. This fallback requires download permission, adds download traffic, streams without saving a local copy, and stops after five minutes. Unreadable or different contents remain Failed; no upload retry or remote deletion occurs. Results say whether size or downloaded contents were verified.

Certificates must be trusted and valid. Redirects are refused; save the final HTTPS URL. Websites with upload forms, multipart POST APIs and signed query URLs are not supported by this adapter. A WebDAV endpoint supporting these same HEAD and conditional PUT operations may work. Replacement can also read WebDAV file-version metadata; directory creation and other WebDAV file operations are not implemented.

Replace existing HTTPS files

Replace Existing HTTPS Files is a separate main-window option, unchecked by default. With it unchecked, existing HTTPS names skip. With it checked, new names still copy using If-None-Match: *. Existing files require a strong **ETag**, a version identifier supplied by HEAD or, when HEAD cannot supply one, a read-only WebDAV PROPFIND metadata request. This fallback checks only that exact file, requires successful version/type properties for an ordinary file, and is limited to 64 KiB and 30 seconds. Missing/weak identifiers, collections, ambiguous metadata or unsupported endpoints block replacement before uploading; neither a connection test nor an ordinary upload proves replacement support.

Preview reports **Would replace** when the existing file supplies a usable identifier; it makes no changes. Run Now checks again and sends PUT with If-Match containing the version checked during that run. The server must honor that condition: if another program changes/removes the file between the check and upload, HTTP 412 records Skipped with a conditional-replacement explanation. Preview again before another attempt. The app does not retry the upload or weaken the condition. A provider that ignores conditional headers cannot provide this protection. The protocol contract is described in [HTTP conditional requests](#).

Every successful replacement requires an authenticated download and a byte-for-byte comparison with the source, even if HEAD reports a matching size. Results and History show **Replaced** only after that comparison succeeds. Download permission is required and verification adds download traffic with a five-minute read-back deadline. The source remains in place; no backup is created. No remote DELETE, delete-then-upload fallback, or automatic upload retry is performed.

HTTPS replacement sends PUT directly to the file URL. Preservation of the old file during an interrupted upload depends on the endpoint's storage behavior; a failed or interrupted request can leave updated or partial contents. Inspect the destination before retrying any uncertain result. A successful content check does not independently prove how the server handles simultaneous writers or failed uploads.

The choice saves with current selections and feeds, restores with Use Feed, and applies to schedules. Clear Selections/default clear-after-save turn it off. Updating a feed disables its schedule; review/re-enable it in Saved Feeds → Schedule. Existing saved feeds default this new option off.

Verify HTTPS replacement on Files.com or another compatible endpoint

1. Rebuild and start the application. Use a source folder containing only a fresh `https-replace-check.txt` with First version inside.

2. Clear Selections, choose that source and add only the saved HTTPS destination. Leave all replacement checkboxes unchecked.
3. Preview and Run Now: expect Would copy and Copied. Refresh the provider's website, download a fresh copy and confirm First version.
4. Change the source to Second version, keeping its name. Preview and run: expect Would skip and Skipped. A fresh download should still contain First version.
5. Check Replace Existing HTTPS Files. Preview should show Would replace, while the downloaded destination still contains First version. If it reports a missing version identifier, stop and share the details; this endpoint cannot currently use conditional replacement.
6. Run Now: expect Replaced. Refresh/download again and confirm Second version; confirm the source still contains Second version.
7. Check History for Replaced and details indicating downloaded contents were verified.
8. Save a disposable feed with clear-after-save checked; confirm clearing resets the checkbox, then Use Feed restores it. Restart and confirm current selections retain the choice.
9. Change the source to Third version, enable a short interval schedule for that feed, close the dialog and leave the app open. Download afterward and confirm Third version with Scheduled/Replaced History. Disable the test schedule.

Stop at any failed/uncertain step and inspect the destination before retrying. A live ordinary replacement test does not also establish concurrent-write or interrupted-upload guarantees.

If the response is lost or size verification fails, a file may already have been stored. The app reports the uncertainty and does not automatically retry or delete it. Check the destination before retrying.

The temporary localhost HTTPS test server used during development was removed after verification.

<https://localhost:7443/uploads/> is not a production destination and will not work without a server running there.

An obsolete **Local HTTPS test** profile may be deleted through Saved Locations.

Schedule a named feed

This version supports **Interval** schedules from 1 to 10080 minutes (one week), **Daily** schedules at a chosen time and time zone, **Weekly** schedules on selected weekdays, and **One Time** schedules for a specific date and time. Keep the computer awake. The app must remain open unless you enable **Background Runs** for closed-app scheduling while signed into Windows. New feeds have no enabled schedule.

Enable automatic runs controls the individual feed; **Background Runs** controls whether enabled feeds can also run with the application closed. Saved Feeds does not need to remain open. Close that window after saving a schedule; leaving the main application open or hidden in the tray permits in-app scheduling.

1. Save and manually test a feed before scheduling it.
2. Click **Saved Feeds...**, select that feed, and review its source and destinations.
3. Click **Schedule...**
4. Choose **Interval**, then select **Every 30 Minutes**, **Every Hour**, **Every 3 Hours**, **Every 6 Hours**, **Every 12 Hours**, or **Custom**. Custom accepts 1-10080 whole minutes. Existing schedules reopen with their matching preset, or Custom when their interval is different.
5. Check **Enable automatic runs**.

6. Click **Save Schedule**. Its first run is due one interval after saving; it does not run immediately.
7. Review the Enabled status and Next run time in the feed details.
8. Close Saved Feeds and leave the main window open. Opening configuration/history dialogs or performing a manual operation delays scheduled runs until the app is free.
9. Watch the schedule status message on the main window. No result dialog opens automatically.
10. Open **Feeds** → **Delivery History** afterward. Scheduled entries identify the feed and show **Scheduled** in the Trigger column. Select the run and use **View Delivery Results** for details.

To disable automatic runs, open that feed's **Schedule...**, uncheck **Enable automatic runs**, and click **Save Schedule**. Saving a schedule always starts a new countdown when enabled. Cancelling the dialog leaves the existing schedule unchanged.

Only one run executes at a time within the app, and only one instance can manage schedules and edit/delete feeds for this Windows user. Other instances can still use feeds and run manually; avoid opening multiple instances for normal operation. If scheduling is unavailable, close all extra instances and reopen one app. A saved feed or settings error can also prevent scheduling.

For Interval schedules, the next interval begins after the previous run finishes. Daily schedules keep the chosen wall-clock time rather than adding 24 hours to the previous finish. If the app was closed, asleep or busy past the due time, the enabled feed runs once when the app is available; missed intervals are not replayed. An overdue enabled schedule can therefore run shortly after reopening the app. Due times are stored with their UTC offset and displayed in your computer's local time. Daily schedules separately retain their selected time zone, even if the computer time zone changes. Changing the computer clock can affect when a run becomes due.

Schedules use the named feed's saved selections and each referenced location's current saved connection settings. They do not use or replace your current main-window selections. Editing a saved location can change where an enabled feed delivers; disable affected schedules while changing connection settings. Replacing a named feed or deleting it disables/removes its schedule.

If any delivery fails after the configured connection checks, a feed cannot be resolved, or saving its history fails, that schedule is disabled for review. Check History, the main schedule status and the feed's Last result before re-enabling. Errors before a delivery history record can be created appear in the schedule status/Last result. A schedule interrupted by a crash is disabled when the app restarts; inspect destination files before re-enabling. The app attempts to persist these failure states; if persistence fails, scheduling stops in that instance and reopening requires another review.

Connection retries and recovery

Folder-source feeds offer **Extra connection checks after a temporary failure (0 = off)** in the Schedule window. Outlook mailbox and OneDrive feeds require zero automatic connection retries; their retry choice is disabled. Select 0, 1, 2 or 3 extra checks. The default is 0, including existing feeds. Set **Wait between connection retries, in seconds (5-300)** when retries are enabled; the initial default wait is 30 seconds. Save Schedule to apply. These settings apply to scheduled runs; ordinary Preview and Run Now do not use them.

Retries only repeat failed read-only checks for remote destinations before any files are delivered. Recognized network connection/DNS failures, timeouts and HTTP 408/502/503/504 responses qualify. Other HTTP errors, authentication failures, certificate/host-key trust failures and configuration errors require review. Local-folder failures do not automatically retry. A retry checks the destination name again; if it now exists, the file is skipped. Already successful checks are not repeated.

The main schedule status displays the wait and retry number. The app stays busy during the checks and waits, so other runs and configuration actions wait until this attempt finishes. Healthy destinations are delivered after the connection

checks finish, even if another destination remains unavailable. History's delivery Details records how many extra checks were attempted for each affected file. These checks belong to one scheduled run, not separate history entries. Connection operation timeouts add to the configured waits.

Upload failures are never automatically repeated, even when retries are enabled. A lost response can mean a file was stored already. Exhausted connection checks or any failed delivery disable the affected schedule as before. One Time remains disabled after its single attempt, including connection retries within that attempt. Exiting/crashing during an attempt requires review after restart; it does not resume the retry sequence automatically. There is no background retry queue after the app exits.

To recover a disabled feed:

1. Inspect **History**, the feed's **Last result**, and the destination. If publication/storage is uncertain, determine whether the file arrived before attempting another delivery.
2. Correct the network, saved connection, permissions or source problem. Use **Test Saved Connection** for remote settings; a successful connection test does not prove upload permission.
3. Select the feed and click **Use Feed**, then **Preview** to check its current destination names.
4. When ready, reopen its Schedule, check **Enable automatic runs**, and save. A One Time schedule needs a new future date/time. Re-enabling starts a new scheduled attempt; it does not resume individual files from the failed run. Existing names follow the source workflow: folder feeds use the saved replacement choices, Outlook attachments use delivery records and stable names, and OneDrive uses its automatic newer-file rule.

For a disposable retry-limit test, save an HTTPS location called Retry Test with URL

`https://127.0.0.1:65432/uploads/`, authentication None, and no server listening on that port. Save a separate feed from a disposable source containing one file to this location. Configure Custom one minute, two extra checks and a five-second wait, enable/save, then close Saved Feeds. Within about 90 seconds, the connection should fail after two retries; History should show Failed with `retried 2 time(s)`, and the feed should be Disabled with no Next run. No files should be uploaded or removed. Reopen Schedule to confirm the retry settings persist, then restart to confirm Disabled persists. Delete the disposable feed and location after testing. If that port hosts another service or the failure is a TLS error instead of connection refused, choose another unused local port.

Feed storage now writes schema version 8 and reads versions 1-8. Older builds refuse version 8 rather than silently losing settings; restoring older source checkpoints may also require restoring compatible feed data.

Daily schedules

1. Open **Saved Feeds...**, select the tested feed and click **Schedule...**
2. Choose schedule type **Daily**.
3. Enter a time such as 9:30 AM, 2:30 PM, or 14:30.
4. Select the time zone. New daily schedules default to the computer's current time zone; an existing daily schedule retains its saved zone.
5. Check **Enable automatic runs**, then click **Save Schedule**.
6. Review the chosen daily time, saved zone and Next run in the feed details. Next run is displayed in the computer's local time.
7. Close Saved Feeds and leave the app open and computer awake.

Saving selects the next occurrence strictly after the save time. If today's chosen time has passed, Next run is tomorrow. Daily schedules are calendar times in the selected zone, not a fixed 24-hour interval. A scheduled run delayed by a busy

app or sleep runs once when available, then recalculates the next future daily time; missed days are not replayed.

During a spring-forward clock change, a selected time that does not exist runs at the first valid minute after the gap. During a fall-back change, a repeated time uses its first occurrence; it is not scheduled again for the second occurrence. The calculation uses the selected zone's rules. Clock changes, delayed runs and the normal interrupted-run protections still apply.

To test without waiting a day:

1. Use a disposable local feed and add a fresh filename to its source.
2. Set Daily to about two minutes ahead of the current time in your selected zone and enable it.
3. Save, confirm Next run is today at the expected time, and close Saved Feeds.
4. Leave the main window idle without clicking Run Now. Wait until about 10 seconds after the chosen minute.
5. Confirm the file was copied and History shows the feed name and Scheduled trigger.
6. Open Saved Feeds and confirm the next run is the following day's chosen time. Confirm your main-window selections were not replaced by the scheduled run.
7. Disable and save the schedule, then close/reopen the app. Confirm the Daily type, chosen time and zone remain saved, with the schedule Disabled.

Weekly schedules

1. Open **Saved Feeds...**, select a tested feed and click **Schedule...**
2. Choose schedule type **Weekly**.
3. Enter the run time and select its time zone.
4. Check one or more weekdays. At least one day is required, even when saving a disabled schedule.
5. Enable automatic runs if desired and click **Save Schedule**.
6. Review the selected days, time, zone and Next run, then close Saved Feeds.

Saving chooses the next future occurrence on a checked weekday in the selected zone. If today's chosen time has passed, it advances to the next checked day; selecting just today may make the next run a week away. After completion, the next future checked day is calculated. The spring/fall clock-change rules described for Daily also apply to Weekly. Overdue runs execute once when scheduling is available; missed weeks are not replayed. Keep the computer awake and either leave the app open or enable Background Runs while signed into Windows.

For a short test, add a fresh file to a disposable local feed, select only today's weekday, and set the time about two minutes ahead. Enable and save, confirm Next run is today, close Saved Feeds, and leave the app idle. Confirm the automatic copy and Scheduled history result. Afterward, confirm Next run is the same selected weekday next week. Disable and save, restart, and confirm type, days, time and zone remain saved. Finally try saving with no days selected: an explanation should appear and the dialog should remain open. Cancel to retain the previous schedule.

One-time schedules

1. Open **Saved Feeds...**, select a tested feed and click **Schedule...**
2. Choose **One Time**.
3. Select the date using the date picker, enter the time and select its time zone.

- 4. Check **Enable automatic runs** and click **Save Schedule**. The selected instant must be in the future when enabling it.
- 5. Check the date/time/zone summary and Next run, then close Saved Feeds.
- 6. Leave the app open and computer awake. After its single attempt, inspect History and the feed's Last result.

The schedule is consumed and disabled before delivery starts; it does not repeat. Completion retains the chosen date/time and Last result for inspection, with Next run shown as None. To run once again, select a new future date/time and explicitly enable it. A saved overdue one-time schedule that has not started attempts once when the app is available. An interrupted attempt remains disabled on restart; check destinations and history before configuring another attempt.

A time missing during a spring-forward clock change is rejected: choose a valid time on that date. A repeated fall-back time uses its first occurrence. Disabling a schedule can retain a past date for reference, but enabling that past date is refused. Cancelling an edit does not change the existing schedule.

For a short test, use a fresh file in a disposable local feed, choose today's date and a time about two minutes ahead, and enable/save. Confirm today's Next run, close Saved Feeds and leave the app idle. Confirm the single copy and Scheduled history entry, then check the schedule automatically became Disabled with Next run None. Add a second fresh filename and wait about 70 seconds: it must not transfer automatically. Restart and confirm Disabled persists. Finally try enabling yesterday's date; saving should explain that a future date/time is required. Cancel afterward.

First interval schedule test

- 1. Use a disposable source folder and a working local destination; save a feed for them.
- 2. Add a fresh file, such as scheduled-check-01.txt, to that source. It must not already exist at the destination.
- 3. Choose Interval → Custom, enter 1 minute, and enable the feed's schedule.
- 4. Close Saved Feeds and leave the main window idle. Do not click Run Now.
- 5. Wait about 70 seconds and confirm the file appears at the destination.
- 6. Open History and confirm the feed name, Scheduled trigger and Copied result. Close History afterward.
- 7. Leave the app idle for another interval; confirm the repeat run is recorded as Skipped.
- 8. Disable the schedule and save, then close Saved Feeds.
- 9. Add a different fresh test file and wait about 70 seconds. Confirm it is not copied while the schedule is disabled.
- 10. Close/reopen the app, select the feed in Saved Feeds and confirm the schedule remains Disabled.

Preview, Run Now and results

Always check the destination list before running. A profile in Saved Locations and a destination selected on the main window are different things.

Result	Meaning
Would copy in Preview	The current checks found no collision. Copying/uploading has not started; write permission may still fail.
Would replace in Preview	A folder-source replacement choice applies, or OneDrive is newer than an existing regular destination file. No changes have been made; server publication support remains untested.

Result	Meaning
Would skip in Preview	The destination name already exists and would be skipped.
Cannot check in Preview	The check failed. Read Details and correct the issue before running.
Copied after Run Now	Delivery succeeded according to the connector's checks.
Replaced after Run Now or in History	The existing destination file was replaced under the source workflow. Read Details for verification and any retained recovery path.
Skipped after Run Now or in History	No delivery was needed or allowed: for example an occupied name, an already delivered attachment, an equal/newer copy of a OneDrive source file, or any existing name at a OneDrive upload destination. Read Details; differing contents alone do not override the rule.
Failed	Read Details for the error. After a connection interruption, the remote result may be uncertain.

Preview is a snapshot. Files or server conditions may change before Run Now; delivery checks again. Results are per file and destination, so a mixed run can contain both successes and failures. Counts include those delivery attempts, not just distinct source filenames.

Local, SFTP and FTP/FTPS delivery use temporary files and size checks. A connection loss can leave temporary files; inspect any cleanup or uncertain-publication message before retrying. Do not delete unfamiliar server files.

Delivery history and Clear History

1. Open **Feeds** → **Delivery History** on the main window.
2. Select a run to see its ID, source and destinations.
3. Click **View Delivery Results** to see each file's Copied, Replaced, Skipped or Failed result and Details.

Click **Copy Run Details** to copy the selected run's ID, feed, times, source/destinations, counts and all delivery messages as plain text. Paste it into a message or document. The history table and results tables wrap long paths and messages.

In **View Delivery Results**, Preview or Run Now results, select a row to read its full information in the wrapped text box below the table. Select text there and press Ctrl+C, or use **Copy Selected** to copy selected rows and **Copy All** to copy every result. Hold Ctrl to select several rows. Copying does not change files or history.

Started and finished times display in your computer's local time. History remains after closing the app. It records paths and outcomes, not a backup of delivered file contents.

To clear records:

1. Open **Feeds** → **Delivery History**.
2. Click **Clear History**.
3. Read the confirmation and choose **Yes** to remove the displayed saved runs, or **No** to keep them.
4. New runs will save history normally afterward.

Clearing history does not delete source files, destination files, selections or saved locations. If clearing partially fails, reopen History to inspect the remaining records. If delivery finishes but saving history fails, review the delivery results; do not rerun merely to create a history record.

Verify a new destination: upload, then skip

Use a disposable source folder containing only one test file. Use a fresh filename. Leave all replacement checkboxes unchecked for this skip test. Do not use another program to upload that same name during the test.

1. Create a file such as `delivery-check-01.txt` containing `First version` and save it.
2. Choose its source folder and **only the destination being tested** on the main window.
3. Click **Preview**. Expect `Would copy`. If it shows a collision, choose a fresh filename and preview again.
4. Click **Run Now**. Expect `Copied`.
5. Inspect the destination: open the copied file for a local/network folder, download it using FileZilla for SFTP/FTP/FTPS, or use the endpoint's download URL/tool for HTTPS.
6. Keep downloaded verification copies in a separate folder so they do not replace your source. Confirm `First version`.
7. Edit the original source to `Second version`, keep its filename unchanged, and save.
8. Click **Preview**. Expect `Would skip (a collision)`.
9. Click **Run Now**. Expect `Skipped`.
10. Obtain a fresh destination copy again. Confirm it still contains `First version`.
11. Confirm the source contains `Second version`.
12. Open History and confirm the second run records `Skipped` for the correct destination.

If the endpoint denies downloads, you can check upload and skip results, but cannot confirm unchanged contents by downloading. Do not describe that as complete contents verification. This test checks ordinary collision handling; it does not prove that an FTP server enforces overwrite rejection during simultaneous uploads.

Restart verification

1. Note the source and selected destinations on the main window.
2. Close and reopen the application.
3. Confirm those selections remain.
4. Open Saved Locations and confirm profiles remain.
5. Open History and confirm earlier runs remain.

No upload is needed for this check.

Troubleshooting

Symptom	Steps to take
Password/token is blank on reopening	Read the saved-credential message. Leave blank to retain it and use Save and test. Re-enter only for replacement or when the saved credential is missing.
Test succeeds, upload fails	The test is read-only. Check write/rename permissions and the delivery Details.
SFTP key inspected but not trusted	Verify its fingerprint independently, save the verified value, and test again.
SFTP fingerprint mismatch	Verify the server identity/key change with its operator before updating the saved fingerprint.
FTP/FTPS upload disabled	Read and choose a profile's upload protection option. Do not claim administrator confirmation without confirmation.
FTP/FTPS folder/listing or data connection fails	Check the exact protocol, security mode, port, folder and Passive/Active setting. Use the reported stage and FTP response code.
FileZilla download returns 550 Permission denied	The server rejected downloading. Read permissions shown by FileZilla do not prove account-level download access. Check provider restrictions; successful uploading does not guarantee downloading.
HTTPS returns 401 or 403	Check authentication and permissions. HEAD, PUT and download permissions may differ. A denied HEAD is not proof that a file is absent.
HTTPS returns 405 or 501	The endpoint does not support the required HEAD or PUT operation. Obtain a compatible endpoint.
HTTPS reports a redirect	Enter its final HTTPS URL. The app does not follow redirects.
HTTPS TLS connection fails	Check the hostname, network and certificate trust/validity. Do not bypass validation.
Browser gives 404 after a reported upload	Check History's exact destination, the actual filename including extensions, and the full URL. Confirm the upload went to HTTPS rather than another selected server.
Would copy followed by Skipped	A destination name may have appeared after Preview. Inspect the destination.
Failed result says publication/storage is uncertain	Inspect the destination before retrying. The file may have been stored despite the lost response.
OneDrive source changed but delivery skips	Wait for synchronization and compare source/server UTC timestamps in Details. The source must be later than the destination; remote comparisons use whole seconds.

Symptom	Steps to take
OneDrive file arrived but result is Failed	Verification may have failed after publication. Inspect contents and Details before retrying; a recovery path may identify the older file.
OneDrive/Outlook schedule retries cannot be changed	Microsoft-source feeds require zero automatic retries. Reconnect manually if silent sign-in needs renewal.
Enabled feed does not run while the app is closed	Enable/update Setup → Background Runs as well as the feed schedule. Keep Windows signed in and awake; review Saved Feeds and History for disabled or failed runs.
Local folder is rejected	Check that it exists, is accessible, does not overlap another selected folder, and does not use a link/junction.

When seeking help, provide the protocol, sanitized destination, exact result/error, and whether the problem occurred during connection testing, Preview, upload or download. Do not send passwords, tokens or private-key contents.

Stored data

Settings and history are kept for your Windows user under:

```
%LOCALAPPDATA%\NM Softworks\AutomatedFileRouting\
```

This contains `folder-settings.json`, `locations.json`, `routes.json` and the `History` folder. Credentials are stored separately in Windows Credential Manager, not in these JSON files. Copying the JSON files to another user or computer does not transfer the credentials; they must be entered there separately. Private-key profiles refer to a key file on disk, which must remain available at its saved path.

History and profile files include server names and paths. Handle copies according to your organization's needs. Use the application's Delete and Clear History controls for routine cleanup.

Verification coverage

Development checks and user testing have confirmed local/network delivery behavior, saved locations and credentials, SFTP upload and unchanged-content collision checks, FTP/FTPS connections plus upload/skip/history behavior, local HTTPS PUT upload and unchanged-content collision checks, restart persistence, Clear History, and the SFTP Save and test workflow.

The user verified plain FTP upload/skip behavior and unchanged destination contents on a working external test server. The user also verified live plain FTP replacement: the downloaded destination contained Second version, the source was retained, and History reported Replaced. The user verified saving and clearing the replacement choice, Use Feed restoration, restart persistence, and scheduled FTP replacement with updated downloaded contents and Scheduled/Replaced History. Simulated FTP/FTPS checks passed. The user also verified live explicit FTPS connection, first upload, default same-name skip with unchanged downloaded contents, opt-in replacement with updated downloaded contents, retained source and Replaced History on an external SFTPCloud test account. Earlier generic SFTP replacement was refused by the tested server and retained the original file. The later OneDrive SFTP protected-replacement workflow and OneDrive FTP/FTPS manual/scheduled/background workflows passed user tests. Implicit FTPS, live HTTPS Bearer authentication and additional third-party endpoints still require their own environment-specific verification; protocol-level pass reports do not establish every server mode. The user verified external Files.com HTTPS Basic upload and post-upload

verification, downloaded contents, changed-source/same-name skipping, source retention and Skipped History. HTTPS conditional replacement passed simulated checks and live Files.com replacement testing: the updated destination contents, retained source and Replaced History were confirmed using WebDAV file-version metadata. The user also verified HTTPS replacement feed saving and clearing, Use Feed restoration, restart persistence, and scheduled replacement with freshly downloaded Third version and Scheduled/Replaced History; simultaneous-write and interrupted-upload guarantees require separate endpoint-specific checks. The local HTTPS helper has been removed. Provider-specific compatibility should be checked using the numbered destination verification procedure before routine use.

OneDrive: copy to an existing local folder

In Setup > OneDrive Source Accounts, select the saved personal or work account and its OneDrive folder, then choose Copy To Local Folder. Choose an existing local destination. The app copies ordinary files directly in the selected folder; it does not descend into subfolders. Original names are retained. Same-name regular files are replaced only when the OneDrive modification time is newer. Equal/newer destination times skip, and folders/links are not replaced. Local destination modification times preserve the OneDrive source time. There are no delivery receipts: removing a destination copy yourself lets the next manual run copy it again. OneDrive originals remain unchanged.

The direct Copy To Local Folder action does not create a saved feed, History entry or schedule; use the main-window feed workflow for those features. It uses the existing Files.Read permission. Limits are 1 GiB per file and two minutes for the complete operation; use manageable folders within these limits. Linked/redirected local folders and invalid Windows filenames are refused. A failed, cancelled or timed-out run retains completed copies and stops remaining work; review the reported counts and destination before retrying. Incomplete temporary downloads are cleaned up during normal cancellation/failure handling. A process crash or power loss can leave a uniquely named .afr-*.tmp file; it is not a completed destination copy. Source metadata and version must remain unchanged through each download before publication.

OneDrive sources in saved feeds

On the main window, choose Use Saved, select a saved personal/work account and folder, then Use As Source. Add existing local destination folders or saved SFTP/FTP/FTPS/HTTPS destinations and use Preview, Run Now or Feeds > Save Feed. Preview reads metadata only. Run Now records copied/skipped/failed outcomes in History. Loading a saved feed uses the account, drive and folder captured when it was saved, even if another account or folder is now selected in OneDrive Source Accounts. Display paths can become stale after remote renames; saved IDs identify the folder. A disconnected account must be reconnected before the feed can run.

Feeds support manual and scheduled runs to one or more existing local folders and saved SFTP/FTP/FTPS/HTTPS destinations, including mixed feeds. OneDrive originals remain unchanged; direct files only; newer source timestamps replace older same-name regular files. Equal/older source timestamps skip. No archiving or delivery receipts; replacement creates no backup. Removing a local copy yourself allows another manual run to copy it again. Limits remain 1 GiB per file and two minutes per complete feed run. A failure stops remaining work and History retains outcomes for completed copies plus the failure. Preview does not guarantee the later run will succeed: source versions and destination names are checked again during delivery. OneDrive Schedule is available in Saved Feeds; FTP/FTPS destinations also support OneDrive scheduling and Background Runs. Other source types retain their existing scheduling support. Smaller main windows scroll so source choices, destinations and bottom actions remain accessible.

Saved selections now use schema 11 and saved feed documents schema 13. Earlier supported formats still load; older application builds cannot read the new formats. Source checkpoints do not roll back application data.

The main window opens as a taller, normal resizable window, capped at the available screen height. History, Preview and Run Now stay at the bottom while the content above scrolls when needed.

OneDrive to SFTP

Select a OneDrive feed source, then Add Saved Location and choose a working SFTP destination. The location needs its verified SHA256 host fingerprint and current saved credentials. Reuse existing test folders and files; expired test-server credentials need replacing before live testing. Preview checks OneDrive metadata and SFTP destination names without downloading source contents or uploading files. Newer OneDrive timestamps replace older same-name regular files; equal/older source times and remote folders/links skip. Original filenames are kept. Run Now checks the source version/metadata and exact download size into a private temporary local copy before calling the existing SFTP delivery path. SFTP uploads use an exclusively created temporary remote file and regular-file type, size and UTC timestamp verification. Source time is requested; a usable server time at least as new as the source is accepted and reported. New names use non-overwriting rename; replacement prefers POSIX rename, with protected standard-rename replacement only after an explicit unsupported response. No delete-then-upload fallback is used. This is size verification, not remote checksum read-back.

Normal completion/failure/cancellation removes the current local staging file and empty directory, and the SFTP path attempts cleanup of its own remote temporary file. Cleanup problems are reported. Abrupt process exit can leave staging files. Completed destination copies remain when later work fails. Publication that cannot be confirmed is reported as a failure for review; there is no automatic retry or deletion of a possibly published final file. Manually removing an SFTP copy lets a later manual run copy it again; no delivery receipts are used. Mixed feeds process local destinations, SFTP, HTTPS, then FTP/FTPS, stopping after a failure. Limits remain 1 GiB per file and two minutes per complete feed run; synchronous SFTP calls have their existing connection/operation timeouts. OneDrive files are never uploaded back, renamed, archived or deleted by this feature.

OneDrive newer-file replacement rule

OneDrive delivery automatically copies absent files, replaces same-name regular files only when the OneDrive modification time is newer, and skips equal/older source times. This applies to Copy To Local Folder and manual or scheduled saved feeds with local/SFTP/FTP/FTPS/HTTPS destinations, including existing saved OneDrive feeds. Generic replacement checkboxes remain disabled for OneDrive; its automatic timestamp rule is shown separately in the policy text. Other source workflows keep their existing rules. Preview reports Would Replace and History reports Replaced.

Local staging preserves the OneDrive UTC modification time before atomic file replacement. SFTP compares whole UTC seconds and requests source-time preservation. A server that silently retains its own timestamp is supported when the returned upload/final timestamp is at least the source time; replacement uses POSIX rename when supported, with protected standard-rename replacement when the server explicitly reports POSIX unsupported. Equal times within the same SFTP second skip. A missing/unusable source timestamp fails safely; an unknown SFTP destination timestamp also fails. Directories and links are not replaced. File size/content differences alone do not trigger replacement. No permanent backup is made after successful verified delivery. Protected SFTP/FTP/FTPS replacement temporarily retains the older file and reports any recovery copy left for review. Remote OneDrive originals remain unchanged.

Source metadata/eTag/size checks remain in place. The destination's recorded time/size is checked again immediately before replacement; detected changes stop delivery. These storage operations do not provide an atomic compare-version-and-replace guarantee against another program changing the destination at that final instant, so coordinate other writers to the same names. Server failures or uncertain publication are reported for review without automatic retry or deleting the final file.

SFTP timestamp checks reread file attributes after setting the temporary upload time, rather than relying only on directory metadata. If verification fails, the message distinguishes missing/unsupported file details, size mismatch, or requested versus reported UTC timestamp. Existing final files remain protected; a differing timestamp is reported in the delivery

result when the server keeps its own valid upload time. Missing timestamps or upload/final times older than the source are refused.

SFTP servers that return their own upload time after a timestamp-setting request are supported. The result shows the source and actual server UTC times; later newer-file comparisons use the destination's actual server timestamp. The app does not keep a separate per-file source-version receipt. A source edit whose timestamp is equal to or older than that actual server time skips, as specified by the newer-file rule. Source time remains preserved for local destinations. SFTP publication verifies temporary and final regular-file type/size and a usable timestamp at least as recent as the source; an unverified published final file is retained for review without automatic retry or deletion. Timestamp-setting operations that raise an error still stop the transfer.

SFTP Preview's Would Replace means the source is newer; Preview does not perform uploads/renames or prove replacement permission. Delivery prefers POSIX replacement. Only an explicit unsupported response permits protected replacement using standard no-overwrite renames: move the older final file to a unique .filerouting-previous-... recovery name in the same folder, publish the checked upload, verify the final metadata, then remove the older recovery file. This alternative has a brief interval with no file at the final name and is not atomic. It is never used after a timeout, permission denial or generic/uncertain POSIX failure.

If protected publication fails, the app attempts to restore the recovery file to its original name only when no final file exists and recovery type/size are valid. It never overwrites a concurrent final file during recovery. Otherwise it keeps the previous file under the recovery name and reports the path for review. Confirmed delivery with recovery cleanup denied reports Replaced plus the retained recovery path. Recovery may also remain after process interruption; there is no automatic deletion of old recovery files on later runs. Server-side rename can change restored timestamps, so review the timestamp before retrying a restored file. Detailed errors identify the operation and SFTP status without exposing sign-in secrets.

OneDrive to HTTPS and scheduled runs

Choose Use Saved on the main window, select the saved account/folder, then Use As Source. Add your existing HTTPS destination with Add Saved Location. Local, SFTP, HTTPS and FTP/FTPS destinations can be mixed; processing order is local, SFTP, HTTPS, then FTP/FTPS, stopping after a failure. OneDrive source originals remain unchanged and subfolders are excluded. FTP/FTPS destinations are also supported for OneDrive sources; see the FTP/FTPS section below.

Preview reads OneDrive and destination metadata only. An absent HTTPS filename is planned as Would Copy; an older same-name HTTPS file is Would Replace; an equal/newer destination skips. Time comparisons use whole UTC seconds. Existing files require Last-Modified and a strong ETag for replacement. If HEAD lacks either, bounded WebDAV PROPFIND metadata can supply a matched file timestamp/version together; an old HEAD timestamp is never paired with a newer PROPFIND version. Unsupported, ambiguous, collection or missing metadata fails before upload. Saved Basic/Bearer/anonymous HTTPS authentication is reused separately from Microsoft sign-in. Redirects are not followed.

Run Now downloads and checks the OneDrive file into local staging, then checks destination metadata again. Conditional PUT uses If-None-Match: * for new filenames or If-Match with the compared ETag for replacement. HTTP 412 reports a skip with no automatic retry. Every successful upload is downloaded and compared byte-for-byte with the staging file. Final Last-Modified must exist and be at least the source timestamp; the endpoint may keep its own upload time, which is used in future comparisons. The endpoint must honor the conditional headers. No backup or remote deletion is performed. Uncertain publication/verification failure is reported for review; destination contents may have been stored. Limits remain 1 GiB per file and two minutes per complete feed run.

Save the feed, then open Saved Feeds > Schedule. OneDrive feeds now share existing interval, daily, weekly and one-time scheduling. Automatic connection retries are fixed at zero. Enabled feeds run while the app is open/in the tray; fully closed-app runs additionally require Setup > Background Runs enabled/updated, Windows signed in, the computer

awake and silent OneDrive sign-in available. No browser sign-in opens during scheduled runs. Failure/interruption disables the affected schedule for review. Saving over a feed disables its schedule as before; review and enable it again. History identifies scheduled runs and copied/replaced/skipped/failed outcomes. Disable test schedules and Background Runs after testing and before rebuilding.

User testing passed OneDrive-to-HTTPS copy, unchanged skip, later edit/replacement, saved-feed restart, personal/work sources, mixed destinations, in-app schedules and fully closed-app Background Runs. The subsequent combined OneDrive FTP/FTPS test round also passed. No settings/schema or Microsoft permission change was needed; older builds that do not support enabled OneDrive schedules should not be used with these newly enabled feeds.

OneDrive to FTP/FTPS, schedules and Background Runs

Add existing saved FTP or FTPS destinations to your OneDrive feed. Preview, Run Now, Save Feed, History, interval/daily/weekly/one-time schedules and closed-app Background Runs use the same automatic newer-file rule. Generic replacement checkboxes stay off for OneDrive. OneDrive originals are retained; no archiving or delivery receipts. Multiple FTP/FTPS endpoints and mixed local/SFTP/HTTPS feeds are supported, stopping remaining work after a failure.

The saved FTP/FTPS location must allow uploads with filename checks. A saved server policy that rejects overwrites conflicts with automatic newer-file replacement and is refused. Do not remove a valid server-policy confirmation just to bypass this check. Existing saved host, port, active/passive mode, explicit/implicit FTPS mode and separate saved password are used. FTPS still requires trusted TLS and encrypted data connections; FTP is unencrypted.

OneDrive comparisons use whole UTC seconds from the server's MDTM command, never ambiguous directory-listing dates. Missing/unusable timestamps or non-regular targets fail safely. Missing filenames copy; older destinations replace; equal/newer destinations skip. Preview reads metadata only and cannot establish write/rename permission.

Each upload is staged under a unique remote temporary name. Its regular-file type, size and SHA-256 content read-back are verified. The app requests source UTC time with MFMT; an explicit unsupported-command reply permits actual server time instead. Other timestamp-setting errors fail. The resulting server timestamp must be usable and at least as new as the OneDrive source; future comparisons use that actual timestamp. The destination's size/time are checked again before publication.

For replacement, the old file moves to a unique .filerouting-previous-...tmp recovery name. The staged file then moves into the final name. Final contents and size must match and its usable UTC timestamp must be at least as new as the OneDrive source before the recovery copy is removed. The server may change modification time when renaming staged or recovery files; exact staged/final timestamp equality is not required. Recovery contents are verified against the older file before accepting its post-rename metadata. On failure, a verified recovery copy is restored only if the final name is absent; otherwise it is kept and its path reported. No final-file deletion or automatic upload retry occurs. A crash can leave recovery/temporary files requiring review. FTP filename checks and rename do not provide atomic protection against other writers; the saved checked-upload setting acknowledges this limitation. The final name is briefly absent during protected replacement.

Schedules require zero automatic connection retries and silent Microsoft sign-in. Enabling a feed runs it while the app is open/in the tray. Enable or update Background Runs after rebuilding for fully closed-app execution while Windows is signed in and awake. Resaving a feed disables its schedule; review and re-enable it. Failed/interrupted runs disable the schedule for review. The existing 1 GiB/file and two-minute feed deadline still apply, with bounded cleanup afterward.

The combined personal/work OneDrive FTP/FTPS manual and scheduling/Background Runs test round passed user verification after the rename-timestamp adjustment. The repository reference ONEDRIVE-FTP-FTPS-TESTS.md retains the completed checklist. Live results apply to the tested environments and are separate from isolated checks.