

Administrative Policy for LocalTranscript

LocalTranscript · Documentation for IT and data protection

Status: version 0.6.65, September 2026

This document is aimed at IT departments and data protection officers at public authorities and companies who want to centrally configure LocalTranscript.

1. Purpose

There is only one installation package per operating system – identical for every brand, every profile, every customer. An organisation that wants to enforce different behaviour on certain machines (e.g. offline operation only, no cloud providers, no mobile companion, no automatic updates) does not need a **separate installation package** for this. It is enough to place a single file named `policy.json` on the target machine, at a machine-wide location that ordinary user accounts cannot write to. The application reads this file at startup and locks in the settings named in it – users can override them neither via the user interface nor by manually editing their own settings file.

This administrative policy has existed since version 0.6.65.

2. How it works

The administrative policy is **not a UI feature**, but is anchored in the application logic itself:

- It is reapplied on **every load and every save** of settings – a manually edited settings value or an outdated save operation from the interface cannot override a locked default.
- It is checked at the respective feature itself, not only in the display. An attempt to trigger a locked action (e.g. sending feedback or syncing Markdown export) via a route other than the visible user interface also fails.
- The application's network protection layer additionally blocks the associated connections, independent of the two aforementioned checks.
- The user interface additionally hides locked settings – this is a convenience feature, not a security measure. What matters for an audit is always the actual behaviour (see section 7 "Verification"), not what the interface displays.

A change to `policy.json` takes effect only at the next start of the application. There is no ongoing monitoring of the file and no way to remotely update an already-running instance – the same model that comparable policy mechanisms of other managed desktop applications use.

Unknown keys in `policy.json` are logged and otherwise ignored (forward-compatible: an older application version that reads a newer policy file with additional keys continues to function). A missing, unreadable, or malformed file is treated as "no policy active" and only logged, never as a hard error – the normal case is an unmanaged installation, so a broken file must not lock out part of the user base.

3. Storage locations per operating system

There is exactly one candidate path per operating system – no search across multiple locations, no additional registry lookup in this version (see section 9 "Planned keys"):

Operating system	Path
Windows	%ProgramData%\com.localtranscript\policy.json (falls back to C:\ProgramData if %ProgramData% is not set)
macOS	/Library/Application Support/com.localtranscript/policy.json
Linux	/etc/com.localtranscript/policy.json

Individually branded builds use their own identifier in the path instead of `com.localtranscript`.

Ownership and permissions

- **macOS and Linux:** the file must be owned by `root` and must not be writable by group or "other" – otherwise it is ignored, with a log entry. This prevents an ordinary, non-administrative user account from placing a weakly protected or deliberately malformed file to fake "no policy active." Roll out as `root` (e.g. via an MDM installation profile, a postinstall script running as `root`, or an appropriately privileged configuration management run), file permissions `0644`.
- **Windows:** there is no comparable ownership concept here – enforcement relies entirely on the permissions of `%ProgramData%`, which by default is writable only by administrators and the `SYSTEM` account (ordinary user accounts have read but not write access). These permissions must not be loosened on the `com.localtranscript` subfolder.

4. Complete schema

All fields are optional – their absence means "this organisation has no policy on this," never "turned off."

Key	Type	Effect	Where enforced
offlineMode	Boolean	<code>true</code> = Hard Offline Mode permanently switched on; users can no longer turn it off	Enforced on every load and save of settings; the network protection layer sets the mode at startup independent of the passed value, and refuses any attempt to turn it off again at runtime – with a log entry
allowCloud	Boolean	<code>false</code> = no cloud provider (OpenAI, Anthropic, Infomaniak, Pyannote Precision-2) may be configured or used – even with an existing internet connection and outside Hard Offline Mode	Enforces local providers on every load/save and deletes existing cloud credentials; the network protection layer additionally blocks every connection whose purpose names a cloud provider
autoUpdate	Boolean	<code>false</code> = the automatic update no longer checks and no longer downloads anything; the update channel is fixed to "stable"	Enforced on every load/save; the update feature refuses, with a notice that updates are managed by the organisation

companion	Boolean	false = the mobile companion cannot be activated	Enforced on every load/save of settings
mcp	Boolean	false = the MCP integration (HTTP and stdio) cannot be activated	Enforced on every load/save; additionally, the separate MCP helper process checks the same policy file independently, regardless of whether the main application has already enforced it
feedback	Boolean	false = sending feedback from within the application is refused	Checked before every feedback submission
voiceprints	Boolean	false = no capture, update, or matching of voiceprints – neither live during recording nor after the fact	Checked before every corresponding action; additionally, automatic voiceprint capture is generally switched off
markdownExport	Boolean	false = the Markdown/Obsidian sync cannot be activated or run	Enforced on every load/save; additionally checked before every sync
dictation	Boolean	false = dictation insertion into other applications and saving of dictation snippets are refused	Checked before every corresponding action
skipWizard	Boolean	true = the setup wizard is considered already completed; users land directly in the application	Set on first start, provided no completion date exists yet
allowedHosts	List of strings	Additional hosts that remain reachable in Hard Offline Mode nonetheless (exact match or subdomain – e.g. "models.intranet.example" also allows "cdn.models.intranet.example", not the reverse)	Additionally considered by the network protection layer alongside the loopback/LAN allowlist

The effective state can be read out via a diagnostic feature of the application, which returns all the values named above plus an additional management status; this is only positive if a correctly matched, validly parsed policy file was actually found.

5. Example: "Strictly managed organisation" profile

Profile for an organisation that wants to enforce offline operation, no cloud providers, no automatic updates, no mobile companion/no MCP integration, no feedback, and a skipped setup wizard:

```
{
  "offlineMode": true,
  "allowCloud": false,
  "autoUpdate": false,
  "companion": false,
  "mcp": false,
  "feedback": false,
  "skipWizard": true
}
```

All keys not named (voiceprints, markdownExport, dictation, allowedHosts) remain untouched and fully user-controllable – this organisation has left no policy for them. If an internal model mirror server should also remain reachable in Hard Offline Mode, add e.g. "allowedHosts": ["models.intranet.example"] .

6. Distribution

Intune (Windows), Win32 app, or PowerShell script: a script running as SYSTEM (or a Win32 app with no actual payload, just a detection rule) places the file:

```
$dir = "$env:ProgramData\com.localtranscript"
New-Item -ItemType Directory -Force -Path $dir | Out-Null
Set-Content -Path "$dir\policy.json" -Value $PolicyJson -Encoding utf8
# The default permissions of %ProgramData% already deny write access
# for ordinary user accounts; no further steps are necessary.
```

A direct OMA-URI/configuration profile for Intune does not yet exist in this version (see section 9) – the route via the Win32 app is the currently supported one.

GPO (Windows): a Group Policy Preferences setting of type "File" that copies policy.json to %ProgramData%\com.localtranscript\policy.json, applied to the **computer**, not the user, so that it lands in the machine-wide %ProgramData%, which ordinary user accounts cannot write to.

macOS MDM: either roll out the file directly via the MDM's file management profile to /Library/Application Support/com.localtranscript/policy.json (root ownership, permissions 644), or use a postinstall script that runs as root during installation of a .pkg package and writes the file. The macOS "Managed Preferences" mechanism (.plist) is not yet read in this version (see section 9).

Linux (configuration management – Ansible/Puppet/Salt, etc.): ensure that /etc/com.localtranscript/policy.json exists, is owned by root:root, permissions 0644. An example for Ansible:

```
- name: Roll out LocalTranscript administrative policy
  ansible.builtin.copy:
    src: policy.json
    dest: /etc/com.localtranscript/policy.json
    owner: root
    group: root
    mode: "0644"
```

7. Verification

Four independent ways to verify that a policy is actually in effect:

1. **Status display in the application.** Settings → System & Support → Diagnostics shows a notice that the installation is managed by the organisation, as soon as a policy file has been found, correctly matched, and successfully read. This is the fastest check that the file is in the right place with the right permissions – an incorrect path or incorrect permissions otherwise silently fall back to "no policy" and appear only in the technical log, not as an error message in the interface.
2. **Network audit log.** Settings → System & Support → Diagnostics → Network Log records every refused connection with the reason "policy" (offline mode locked) or "cloud providers disabled by policy" (cloud providers locked by policy) – this is the evidence usable for a procurement review (see `network-disclosure.md`).
3. **Direct feature invocation instead of only looking at the interface.** Since the interface hiding is purely cosmetic, a security-critical review should trigger the underlying feature directly (e.g. attempting to send feedback via the application's technical interface). This must be rejected with a clear error message "This feature is managed by your organisation." as soon as the corresponding key is set to `false`.
4. **Manually edited settings file.** A manually entered, actually prohibited change in `app-settings.json` (e.g. a cloud API key or `offlineMode: false`) must be overwritten again at the next start of the application – the policy is applied on every load of settings, before the application proceeds to operate with these values.

8. What users can still change

Everything that is **not** named by a set key. A policy that only sets `allowCloud: false` makes no statement about the mobile companion, dictation feature, or setup wizard – these remain exactly as in any unmanaged installation, fully controllable by the user.

The policy acts exclusively as a restriction: it can turn features off (or enforce Hard Offline Mode, or mark the wizard as completed), but it cannot turn a feature **on** that a brand profile has already hidden. There is also no key that forces a feature against the will of the user – with the two documented exceptions `offlineMode: true` and `skipWizard: true`, which by their nature act in an enforcing way regardless.

9. Planned keys

The following extensions are planned for a future version, but **not yet shipped in version 0.6.65**:

- **Retention keys** (`retentionMeetingsDays` , `retentionAudioDays` , `retentionDictationDays` , `retentionAuditDays`): retention periods are today purely a **user setting** (see our deletion concept), not yet a policy-enforceable default. An organisation currently cannot prevent a user from leaving all four fields empty ("retain forever"). It is planned to add the same four retention fields as policy keys that – like `offlineMode` – are enforced on every load and save of settings.
- **Additional distribution source: Windows registry** (`HKLM\SOFTWARE\Policies\...`), which would be evaluated before the JSON file.
- **Additional distribution source: macOS "Managed Preferences"** (`.plist`), which would be evaluated before the JSON file.
- **Further keys**: a configurable internal model mirror server (`model_mirror_base`), a central storage directory for audit logs (`audit_dir` , e.g. a network drive), a control for deep links (`deep_links`), and more fine-grained feature control (`features`).
- **Finished distribution packages**: an ADMX/ADML template for the Group Policy management console, an Intune configuration profile (OMA-URI), and a macOS `.mobileconfig` profile. Until then, the routes described in section 6 (Win32 app/PowerShell script, GPO file preference, MDM file management, configuration management) already work completely today.