MICROSOFT GRAPH ACCESS

Microsoft Graph Permissions — Justification and Minimization

Clevermore accesses Microsoft 365 data through the Microsoft Graph API on behalf of each signed-in user. Every permission we request is delegated, requires tenant admin consent, and maps to a specific user-facing feature. This document enumerates each requested scope and explains the feature it powers.

Delegated Only | Admin Consent Required | Least Privilege | Feature-Mapped

Four Principles That Govern Our Permission Set

1. Delegated Only — Never Application-Level

Every permission Clevermore requests is a **delegated** permission, granted only in the context of a specific signed-in user. We do not request, and do not hold, any *application* (app-only) permissions against your mailbox data. This means Clevermore can never act outside an authenticated user's session, and can never access data the user themselves could not access.

2. Admin Consent Required

All permissions in our app registration are configured to require tenant admin consent. Users cannot self-consent to expand Clevermore's access. A tenant admin reviews the complete permission list once, grants it tenant-wide (or to a subset of users via Entra app assignments), and the grant remains revocable at any time from the Azure portal.

3. Least Privilege — The Narrowest Scope That Works

When Microsoft Graph offers a narrower and a broader version of the same capability, we request the narrower one. Examples in our actual permission set:

- **Files.ReadWrite.AppFolder** instead of **Files.ReadWrite** or **Files.ReadWrite.All** — we can only read and write inside Clevermore's own folder in the user's OneDrive, not the rest of their drive.
- **User.ReadBasic.All** alongside **User.Read.All** — we use the basic profile scope wherever a name and email are sufficient.
- **People.Read** instead of broader directory enumeration permissions for participant lookup.

4. Feature-Mapped

Every requested scope corresponds to a specific user-facing feature. The inventory below lists what each scope does and which Clevermore feature requires it. If a scope no longer maps to a shipped feature, it is removed from the app registration.

> **What this combination guarantees.** Because every permission is delegated, requires admin consent, and maps to a documented feature: (1) Clevermore cannot read your tenant's data without a user being signed in; (2) the consent process gives your IT admin a single, auditable point of control; and (3) we cannot quietly grow our access — new permissions require a new admin consent.

Permissions Requested by the Outlook Add-in

The following permissions are declared in the Clevermore Outlook add-in's Entra app registration (multi-tenant, app ID `f72f1cdb-d05f-4202-8b7a-5a192d8b6c15`). Permissions are grouped by capability area.

IDENTITY & SIGN-IN

Standard OIDC scopes for user authentication

These are the baseline OpenID Connect scopes used by every Entra-integrated application. They allow Clevermore to identify the signed-in user and maintain a session.

`openid` · Sign-in — required for OpenID Connect.

`profile` · Basic profile (name, tenant ID) returned in the ID token.

`email` · The user's email address in the ID token.

`offline_access` · Refresh tokens, so users are not prompted to re-authenticate on every session.

`User.Read` · Read the signed-in user's own profile (display name, mail, ID).

MAIL

Read, organize, draft, and send mail on behalf of the user

Clevermore's core capabilities — task extraction, catch-up summaries, wiki/knowledge base, draft assistance — all operate on the user's mailbox. Each scope below corresponds to a documented operation.

`Mail.Read` · Read message bodies, subjects, and recipients to extract tasks, build catch-up summaries, and enrich contact profiles.

`Mail.ReadWrite` · Apply Outlook categories, toggle flags, and mark messages read/unread when the user accepts a Clevermore suggestion.

`Mail.ReadWrite.Shared` · Same operations on shared and delegated mailboxes (executive assistant scenarios, where the user has been granted access to another mailbox).

`Mail.Send` · Send drafts the user has explicitly composed or approved (AI-assisted replies).

`Mail.Send.Shared` · Send from a delegated/shared mailbox the user has been granted send-as permission on.

`MailboxFolder.ReadWrite` · Create the Clevermore folder hierarchy the user filing wizard organizes mail into (one folder per client matter).

`MailboxItem.Read` · Read item-level metadata for reliable thread reconstruction across moves and re-files.

`MailboxSettings.ReadWrite` · Read the user's time zone and signature, and update the mailbox's master category list so Clevermore-applied categories render correctly in Outlook.

CALENDAR

Read and create calendar entries for scheduling context

`Calendars.ReadWrite` · Read upcoming events to surface meeting context inside catch-up summaries and tasks (“you have a call with Acme tomorrow”), and create calendar entries when the user accepts a scheduling suggestion.

CONTACTS

Read and update Outlook contacts to power contact enrichment

`Contacts.ReadWrite` · Read existing Outlook contacts to seed Clevermore’s contact profiles; write back enriched profile data (extracted facts, role/title updates) when the user accepts a suggestion.

PEOPLE & DIRECTORY LOOKUP

Resolve participants on emails to display names and basic profiles

When Clevermore presents an email’s participants, contact list, or knowledge-base entry, we resolve identifiers to display names and basic profile fields. We do not enumerate the directory or read extended attributes.

`People.Read` · The relevance-ranked “people the user works with” list, used for participant suggestions.

`User.Read.All` · Read other users’ basic profile information to resolve display names and titles for participants on the user’s emails.

`User.ReadBasic.All` · The basic-fields subset of the above, preferred wherever name and email are sufficient.

GROUPS

Resolve distribution lists to readable names

`Group.Read.All` · Resolve group display names when emails are addressed to distribution lists or Microsoft 365 groups, so participant lists are human-readable rather than showing raw object IDs.

FILES (APP FOLDER ONLY)

Read and write inside Clevermore's own OneDrive folder — nothing else

`Files.ReadWrite.AppFolder` · Read and write files only inside the dedicated Clevermore application folder in the user's OneDrive. Used for app-specific persistence (e.g. cached templates and attachments). **This scope does not permit access to any other file or folder in the user's OneDrive or SharePoint.**

ONLINE MEETINGS

Reserved for upcoming Teams meeting integration

`OnlineMeetings.ReadWrite` · Will let the user create Teams meeting links when scheduling from inside the add-in. Declared in the current manifest so the feature can ship without requiring a tenant-wide re-consent cycle. Not called by any code path today.

TASKS

Reserved for upcoming Microsoft To Do sync

`Tasks.ReadWrite` · Will synchronize Clevermore-extracted tasks with the user's Microsoft To Do lists, when the user opts in. Declared in the current manifest so the feature can ship without requiring a tenant-wide re-consent cycle. Not called by any code path today.

Consent, Revocation, and Auditability

Granting consent

A Microsoft 365 Global Admin (or Privileged Role Admin) grants consent to the full Clevermore permission set in one of three ways: by signing in to the add-in as the first user of the tenant and accepting the consent prompt; by navigating to the Microsoft admin consent URL for the Clevermore app ID; or by granting tenant-wide consent from *Entra ID → Enterprise Applications → Clevermore → Permissions*. The consent dialog displays every permission in this document — no permission is ever requested at runtime outside that initial grant.

Restricting which users can access Clevermore

Tenant admins can require user assignment on the Clevermore Enterprise Application so that only explicitly assigned users (or members of assigned groups) can sign in. This is independent of the consent grant.

Revocation

Consent can be revoked at any time from *Entra ID → Enterprise Applications → Clevermore → Permissions → Review permissions → Revoke*. Once revoked, Clevermore can no longer issue new tokens, and existing tokens expire within the standard Azure AD lifetime (typically one hour). Removing the Enterprise Application entirely terminates all access immediately.

Auditability

The Clevermore app registration is defined as Infrastructure-as-Code in our source repository (Bicep, with the Microsoft Graph extension), so every change to the requested permission set is a source-controlled commit. The deployed permission set in any environment can be verified at any time via `az ad app permission list` or in the Entra ID portal.

Compliance

Clevermore's integration with Microsoft Graph aligns with the Microsoft Graph API Terms of Service, the Microsoft 365 App Compliance Program guidelines, and GDPR/CCPA data minimization principles. For questions about Graph permissions, contact **security@clevermore.ai**.