Extension Store for authors

The Extension Store is the catalog of extensions for the CAM system. Users find extensions on the store website or on the Extensions tab of the CAM system and install them in one click. This guide takes an author from an empty project to an extension in the catalog, on sale if you wish, and through every version after it.

Writing the extension code is covered by the API documentation and the examples; this guide links to them rather than repeating them (see External references).

Your path

1. Become a developer. Once, in your store account: who publishes, your details, the terms. See Getting started.
2. Build and try it. Start from the template and run the extension in your own CAM system. See Build and test locally.
3. Publish. From a GitHub repository, from an AI assistant, or by uploading a package. See Publishing.
4. Pass moderation. A moderator reviews every version before users get it. See Review, updates and visibility.
5. Sell, if you want to. Set a price and receive your share through Stripe. See Selling and payouts.
6. Keep it alive. Ship new versions, support your users, keep pace with new releases of the CAM system.

Publishing for a company, or together with someone? See Organizations and collaborators. To see the store through your users' eyes, read What your users see.

Contents

1. Getting started — tools, account, becoming a developer, choosing a route
2. Store rules and your obligations — what you agree to, what an extension may and may not do
3. Extension anatomy — files, manifest, entry points
4. Build and test locally — installing your build, debugging, restarts
5. Package requirements — what the store checks and where the card comes from
6. Publishing — the three routes and the publication form
7. Review, updates and visibility — moderation, new versions, My published, hiding
8. Selling and payouts — licence check, prices, Stripe payouts, your balance
9. What your users see — trial, purchase, installation, subscriptions and refunds
10. Organizations and collaborators — publishing for a company and with co-authors
11. Troubleshooting — store messages and what to do about them
12. External references — the store, its terms and contacts, the template, examples and API documentation

Getting started

What you need

- Windows 10 or newer.
- The CAM system with the Extensions tab on its Home page, to try your builds.
- The .NET 10 SDK. Extensions are built for `net10.0-windows`.
- An account of the CAM system. The store has no separate sign-up: Sign in on the store website takes the same account.
- A GitHub account — only for the GitHub route (see Choose a route).

Become a developer

The store accepts publications only from developers who have accepted its terms. You do this once, in your account, and it covers every route, GitHub included. The wizard saves each step, so you can stop and come back later.

1. Sign in on the store website and open Account settings → Developer. The Publish button in the header leads there too.
2. Click Become a developer.
3. Type. Choose who publishes: A company or Me, as an individual. The agreement is made with the one you choose.
4. Profile. Choose Free extensions only or Free and paid extensions. You can enable paid extensions later (see Selling and payouts).
5. Your details. Name, country and phone come from your account; check them. The email is the one of your account and changes only there. Public developer name is what users see on your extensions: 2 to 80 characters, each word in one alphabet, unique in the store. For a company this step is called Company representative: add your Position and confirm that you may enter into agreements on the company's behalf.
6. Company. Only for a company: the legal name as in the register, country of incorporation, registration number, registered address and country of control. VAT number and website are optional.
7. Developer terms. Read the documents, tick the box and click I Agree. What they ask of you is summarised in Store rules and your obligations.

The Developer section then shows your details and the terms you accepted, with the date and versions; Download record saves this record. Your details are now part of the agreement: to change them, write to the store team.

New versions of the terms are announced in advance, in the Developer section and on the Publish page. A material change must be accepted before your next publication: until you click I Agree again, publishing pauses, from GitHub too. A minor change asks nothing of you.

Read the rules before you start building: some capabilities may be published only with a licence check, some not at all, and an extension may reach users only through the store (see Store rules and your obligations).

Choose a route

All routes end in the same store checks and the same moderation (see Package requirements).

Route	Best for	Where it is described
Publish from GitHub	Most authors: every version is published by a workflow run	Publishing
AI assistant (MCP server)	Creating, checking and publishing from Cursor, Claude Code and other MCP clients	ency-extension-mcp
Upload a .nupkg	Code outside GitHub, a one-off release, a package your own build produces, publishing for an organization	Publishing

To write the code, start from the examples and the API documentation in External references. Before you publish, read the design rules: they describe how an extension looks and behaves inside the CAM system and end with a pre-publish checklist.

First publication checklist

1. You are a developer and have accepted the current terms.
2. The extension follows the store rules: no code from outside its package, disclosed network use and data processing, no misleading use of the product name (see Store rules and your obligations).
3. It works in your local CAM system (see Build and test locally).
4. The package passes the store checks: manifest, marker tag, released SDK, the SDK stamp of a paid extension, `reservedDomains`, a package id of your own (see Package requirements).
5. The card is ready: `readme.md` with what the extension does, how to reach you for support and how fast you answer, and a privacy disclosure if it processes personal data; an icon and at least one picture in `screenshots/`.
6. You know your answer to the reserved functionality question (see Publishing).

Store rules and your obligations

When you click I Agree, you accept the Developer Agreement and the documents incorporated into it. This page summarises what they ask of an author. It is a guide, not a substitute: the documents themselves are binding, the English text prevails, and the links are in External references.

The documents

Document	What it governs
Developer Agreement	The contract: licences, warranties, liability, fees, termination
Publishing Policy	Content, technical and commercial rules for extensions
Schedule A — Reserved Functionality	The capabilities licensed separately, and whether third parties may publish in them
Schedule B — Technical Requirements	Packaging, entitlement checks, security, quality
Schedule C — Enforcement and Appeals	How decisions are taken, explained and appealed
Commercial Terms	Commission, prices, payouts, refunds, taxes for paid extensions
SDK Licence Terms	What the SDK may be used for
Trade Mark Guidelines	How to refer to the CAM system and its maker

The documents may change, with notice by email and on the store website:

Change	Notice
Most changes	At least 15 days
Commission	At least 60 days
A new area in Schedule A	At least 30 days
An area removed from Schedule A	Takes effect on publication
Required by law, or by an imminent security or fraud risk	Can take effect at once

Changes do not apply backwards. A material change must be accepted before your next submission; for other changes, publishing after the effective date counts as acceptance. You may terminate the agreement during the notice period instead.

Who may publish

- Companies and individuals may publish free extensions. Paid extensions are open to companies and to individuals acting for their trade, business or profession.
- Your details must be accurate and complete. Registration includes sanctions and export-control screening; developers established in or controlled from a sanctioned jurisdiction cannot publish.
- Whoever accepts the terms for a company confirms they may bind it.
- You must hold all rights to distribute the extension.
- Tell the store team within 15 days of any material change to your details, including a change of control of your business. Moving your extensions to another party — for example, from you to your company — needs the store's consent.

What an extension must not do

- Infringe the rights of others, including the maker of the CAM system.
- Contain malware, spyware, cryptominers, backdoors or undisclosed code acting against the user.
- Circumvent, disable, weaken, emulate, replace or interfere with the licensing, activation, entitlement enforcement, signing or anti-tamper mechanisms of the CAM system, or help anyone do so.
- Collect or send user data beyond what its privacy disclosure states.
- Misrepresent its origin, functions or compatibility, or suggest that the maker of the CAM system produced, endorsed or certified it.
- Be non-functional, unmaintained, or a duplicate of another of your extensions without a real difference.
- Produce machine-control output that bypasses the validation, simulation and collision checks of the CAM system where it provides them.
- Download, install, load or update code from anywhere other than its own package published in the store.
- Break the law, including export-control, sanctions, competition and consumer-protection law.

The store is the only channel

An extension — any software that loads into the CAM system through its SDK, API or extension interfaces — may reach users only through the store. You must not distribute it in any other way, or offer any other store, repository, installer, loader or update mechanism for such software, or any means of bypassing signature verification.

Registered developers may load their own unpublished builds on their own machines while they develop (see Build and test locally); this must never become a way to hand builds to users. To give an extension

to a closed group only — an internal tool or a build for one customer — ask the store team for a private listing.

The rule does not cover software that does not load into the CAM system, such as a standalone application or a tool that exchanges files with it through import and export. The SDK itself, however, may be used only for software distributed through the store, unless the store agrees in writing.

Free means free

A free extension must be free in substance. You must not charge for it or for any of its functions — through licence keys or subscriptions sold outside the store, payments inside the extension, or a separately sold component without which it does not do what its card says. Paid services that unlock nothing, such as consulting, training or custom development, and voluntary donations are allowed. To charge, sell the extension through the store (see Selling and payouts).

Reserved functionality

Some capabilities of the CAM system are licensed to users separately. Schedule A lists them as Domains: milling of each tier, turning, EDM, nesting, additive, robot programming, machine models, Tuner, Hyper, direct import of native SolidWorks files, and others. The list is descriptive, not exhaustive: a capability that the maker licenses separately is reserved even before it appears in Schedule A.

- Complementary extensions — preparing inputs, processing outputs, automation, integration with other systems, post-processors, feature recognition, libraries, reports and similar — need no check and are welcome. Import of neutral formats such as STEP, IGES and Parasolid, and of native formats of other CAD systems, is open. A post-processor that turns 3-axis data into simultaneous multi-axis motion, however, falls into the Domain of that motion.
- Gated extensions provide a capability inside a Domain. Each such capability must run only for a user who holds the licence Schedule A names for that Domain, checked through the entitlement-verification API of the SDK at every load and before every run. A licence for a lower tier does not unlock a higher-tier capability, and a licence for one Domain does not unlock another. A gated extension is an addition for licensed users, not a way to avoid buying the module; it may be free or paid on top of the user's licence.
- Closed Domains take no third-party implementations: machining simulation and collision detection, NC output verification, licensing and anti-tamper, distribution and loading of components (the store, its payments and signature verification), and cloud accounts. Extensions that complement them remain welcome.

Every submission declares the Domains the extension works in (see Publishing). Review checks what the extension actually produces, not only the declaration.

Not sure whether your idea falls into a Domain? Ask the store team before you build. The written answer comes within 20 business days and binds the store for 24 months, as long as your description was

complete and accurate and the extension is not changed, configured or marketed in a way that would change the answer. Later changes to Schedule A do not reverse it.

Technical requirements

Schedule B sets the technical rules that review checks:

- No outside code. The extension runs only code from its own package and updates only through the store. Downloading data — tool libraries, cutting data, configuration — is allowed if that data is not executed.
- Network. All communication goes over TLS 1.2 or newer, to endpoints you declare.
- Credentials. Tokens and passwords are kept only in the secure storage of the SDK.
- Licence checks. A gated extension checks the licence through the SDK; a successful check may be cached for up to 72 hours on that installation only. When a check fails, the user gets a clear message naming the licence needed, and functions that need no licence keep working.
- Components. Third-party and open-source components are disclosed with their versions and kept free of known critical vulnerabilities. Their licences must not impose copyleft obligations on users or on the CAM system.
- Quality. The extension does not crash or hang the CAM system, adds no more than 500 ms to its start, releases resources when unloaded, lets the user cancel long operations, and does not change settings or files of the CAM system outside its own scope.
- Machine output. Machine-control output passes the validation pipeline of the CAM system where one exists, and is never presented as verified or collision-free on the extension's own authority.
- Text. User-facing text is available at least in English and is kept out of the code, so that it can be translated.
- Compatibility. The extension works on a supported release of the CAM system: the current one or one of the two preceding major releases.

When Schedule B changes in a way that requires changes to published extensions, you get at least 90 days to adapt.

Names and trade marks

- Your card must clearly show you as the publisher.
- Do not build the product name of the CAM system into the name of your extension, company, domain or social handle in a way that suggests it comes from the maker: *Nesting Pro for* followed by the product name is fine; the product name followed by *Nesting Pro* is not.
- Do not use the logo of the CAM system in your icon or branding, and do not imply endorsement or certification without a written agreement.
- You may say truthfully that the extension works with the CAM system, and you must say which licence a gated extension needs.

If you are unsure about a name, ask the store team; naming questions are answered within 5 business days.

Your obligations as a publisher

- End-user terms. Users get your extension under your own licence terms, which must not conflict with the store rules or with the user's licence of the CAM system; without your own terms, the Standard Extension EULA applies.
- Privacy. If the extension processes personal data, publish a privacy disclosure for it, for example in `readme.md`.
- Support. State on the card how users reach you and how fast you answer, and keep to it.
- Maintenance. Keep the extension working with supported releases, and tell the store team at least 60 days before you stop maintaining it.
- Security. Report a vulnerability, data breach or safety-relevant defect to the store's security address within 72 hours of discovery (see Contacts).

Enforcement and appeals

If an extension breaks the rules, the store applies the least intrusive measure that addresses the breach. Stages need not follow in order:

Stage	Measure
1	Written notice with at least 30 days to fix
2	Removal from featured places and search promotion
3	No new installations; existing ones keep working
4	Removal from the store, with at least 30 days' notice
5	Termination of the agreement and closure of the account, with at least 30 days' notice

- Every measure, and every refusal at review, comes with a statement of reasons: the rule, the facts, what change would fix it where one is possible, and how to appeal.
- New installations can be stopped at once, pending investigation, if the extension appears to circumvent licensing.
- The store acts immediately against malware, imminent security risks, licensing circumvention and other ways of installing components into the CAM system, and where the law or a credible infringement notice requires it; the reasons may follow afterwards.

- A decision is never based on an extension competing with the products of the CAM system maker or reducing its revenue.
- Three or more confirmed measures of stage 3 and above for similar breaches within 12 months may lead straight to removal or termination.
- You can appeal any decision within 30 days, free of charge. The appeal is acknowledged within 5 business days and decided within 20 business days after that, by someone senior to the original decision-maker and not involved in it. An appeal against removal or termination filed within 10 days stops the measure until the decision, except for immediate actions.
- Complaints about how the store treats you are acknowledged within 5 business days and answered within 20.

See Contacts for the addresses.

Leaving the store

You may withdraw an extension from new distribution at any time (see Review, updates and visibility) and terminate the agreement with 30 days' notice. On termination:

- your extensions leave the store, and no new installations happen;
- licences users already have keep working, and for 12 months you keep providing them with security and critical-defect fixes, or help the store hand the extension over in an orderly way;
- what the store owes you is paid in the next payout, whatever the balance;
- your account stays open for 90 days so that you can export your data.

Extension anatomy

An extension is a .NET library that the CAM system loads through a manifest. The template already contains every file listed here; this page explains what the store and the CAM system read from them.

Files

File	Purpose
`<Name>.dll`	The extension code. The CAM system creates its objects through the `CAMAPI.ExtensionFactory` class.
`<Name>.settings.json`	The manifest: the library and its entry points. Without it the CAM system does not register the extension and the store refuses the package.
`package.info.json`	Store metadata: package id, version, author, description, category, tags, SDK version, flags.
`readme.md`	The README on the store card: what the extension does and how to call it.
`screenshots/`	PNG, JPG, GIF or WebP pictures for the card. The first one becomes the cover, and the icon unless you provide one (see Package requirements).

Manifest

The manifest is a JSON file next to the library.

name. The library name. Required.

description. The library description. Required.

module_path. The path to the library. Required; the store refuses a manifest without it.

unload_mode. When the CAM system unloads the library: `umJustAfterExecution` (default), `umBeforeProgramFinalization` or `umManuallyFromUI`. Optional.

extensions. An array of entry points. Each element is an object with exactly one key — the entry point type — whose value describes the entry point: `id`, `name`, and the fields of that type, for example `caption`, `hint_text` and `icon_path` for a utility.

The ids in the manifest must match the ids the factory creates. A minimal manifest is in the template (`src/<Name>.settings.json`).

Entry points

The entry point type decides where the extension appears in the CAM system. The store copies the files and registers the extension, but does not run it.

Type	Manifest key	Where it appears
Utility	`utility`	The Utilities palette, opened by the Utilities button on the top toolbar or Ctrl+Shift+E
Operation popup	`operation_popup`	The context menu of an operation in the technology tree
Geometry node popup	`geom_model_node_popup`	The context menu of a node in the geometry model tree
Operation solver	`OperationSolver`	An operation in the project; the extension calculates the operation and its toolpath
Global	`global`	No visible place; runs when the CAM system starts and finishes
CLData converter	`geom_cldata_converter`	No visible place; processes CLData after toolpath calculation
Utility runner	`utility_runner`	No visible place; runs or wraps other utilities

Code skeletons of every type, and further types with their examples, are in the `cam-api-examples` repository (see External references).

package.info.json

The store builds the card from this file together with the package metadata.

packageId. The package id. It identifies the extension in the store; the first publisher of an id becomes its owner (see Package requirements).

version. The extension version, in NuGet format.

description, author. The card description and author.

category. What the extension does. The category list is on the publication form of the store website; a moderator sets the category if you leave it empty.

tags. Card tags. The list must keep the store marker tag (see Package requirements). You can add tags of your own on the publication form.

sdkVersion. The SDK version the extension is built with. The card shows the minimum version of the CAM system from it.

requiresRestart. `true` if the CAM system must be restarted after installation.

licensed. `true` if the extension runs only with a licence from the store: bought, a trial, or given free. Once set, it stays on every later version (see Selling and payouts).

reservedDomains. The reserved functionality declaration; required, `[]` if the extension provides nothing from Schedule A (see Publishing).

Build and test locally

Try every build in a local CAM system before publishing: the store publishes the package exactly as built.

Everything on this page is for your own development machines. Users may get the extension only through the store: do not hand out builds, `.dext` files or feeds to them (see Store rules and your obligations).

Build

The build command and the output folder of a project made from the template are described in the template README, section Local build & try-out. The examples in `cam-api-examples` are built by their own scripts and can also be packed into a `.dext` file.

Install your build

The Extensions tab of the CAM system opens from the Home page. Its Manage page lists the installed extensions, enables and disables them, and installs extensions from files and feeds.

Install from file···. Installs a `.nupkg` or `.dext` package, or registers a `.json` manifest in place. You can also drag a file onto the tab.

Register a manifest in place. Select `<Name>.settings.json` in the build output folder. The CAM system loads the library from that folder, so the next build is picked up without reinstalling. This is the shortest loop while you develop.

Add feed···. Adds a NuGet feed or a local folder with `.nupkg` and `.dext` files. Its extensions appear on the Store page next to the store catalog — use it to check how your package installs and updates.

A `.dext` file can also be installed by double-clicking it.

Debug

1. Start the CAM system with the `/EXTENSION_DEBUG_FOLDER="<build output folder>"` argument. The CAM system loads the extensions from that folder in addition to the installed ones.
2. Attach the debugger to the CAM system process (.NET Core Attach in VS Code, Attach to Process in Visual Studio).
3. Call the extension from its place in the interface (see Extension anatomy).

Debugging setups for .NET, Delphi and C++ are described in the API documentation, section Debugging (see External references).

Where installed extensions live

Each installed extension gets its own folder `extensions\<Name>\` in the user data folder of the CAM system (under `%APPDATA%`). The list of registered libraries is `extensions.json` in the same `extensions`

folder.

Restart and unloading

- An installed extension is registered, and the CAM system loads it on its own. If it does not appear, restart the CAM system.
- If the package has `"requiresRestart": true`, the tab asks for a restart after installation.
- A library that is still loaded cannot be replaced: the copy fails with an access error. With the default `unload_mode` (`umJustAfterExecution`) the library is unloaded after each run; with other modes, close the CAM system before installing the new build.
- If the extension does not appear where expected, check on the Manage page that the library and its entry point are enabled.

Package requirements

The store publishes a NuGet package (`.nupkg`). On the GitHub route the store packs it itself; on the upload route you upload the package your build produces. Every route passes the same checks.

Layout

Path in the package	Contents
`lib/<target framework>/`	The extension `.dll` with its `.pdb` and `.xml`
`build/`	The manifest `<Name>.settings.json`, `package.info.json`, `readme.md`, other files of the extension
`build/screenshots/`	Card pictures
`build/translations/<language>/readme.md`	Translations of the README, next to `readme.md` (see README translations)
`build/<packageId>.targets`	The list of files for the installer

The `.nuspec` must contain the package id and version, and its id and version must match `package.info.json`.

Where the card comes from

Most of the card is read from the package and changes only with a new version. Tags, the icon and the category can also be set on the publication form; what you set there stays on later versions.

Card element	Source
Id, version, authors, link	`.nuspec` (`id`, `version`, `authors`, `projectUrl`)
Name and entry points	The manifest `<Name>.settings.json`
Description	`package.info.json` and `.nuspec`
README	`readme.md` and its translations
Tags	`package.info.json`, plus up to 10 of your own from the form

Card element	Source
Icon	The icon uploaded on the form; otherwise `<icon>` in `.nuspec`; otherwise the picture named by `iconPath` in `package.info.json`, relative to that file; otherwise a file named `icon` or `logo` in the package; otherwise the first picture in `screenshots/`
Cover and screenshots	The pictures in `screenshots/`
Category	`package.info.json`, the form, Change the category in My published, or the moderator
Licence check	`licensed` in `package.info.json` or Paid on the form

Without an icon the card shows the symbol of the entry point, but an icon and card pictures are required by the technical requirements of the store.

README translations

`readme.md` is the original. Put its translations next to it, in `translations/<language>/readme.md`, for example `translations/de/readme.md` or `translations/pt-BR/readme.md`. The language is a two-letter ISO 639-1 code, optionally with a region. The language of the original is `<language>` in `.nuspec`.

The card offers a README language switch, and the store opens the README in the language the user chose in its header, or in the language of the browser; without a matching translation it shows the original. The Extensions tab of the CAM system shows the original.

The GitHub route does not carry `translations/` into the package yet: to publish translations, upload the package.

What else the package carries

Item	When
Icon and card pictures	Always
Your end-user licence terms	Unless you adopt the standard extension EULA of the store
Privacy disclosure	When the extension processes personal data

Item	When
Third-party licence notices	When the package includes open-source or third-party components

Marker tag

The package tags must include `ency-extension`. The store catalog indexes only packages with this tag. The template and `ExtensionManagerCLI pack-library` add it (see Publishing).

SDK version

The extension must be built against a released SDK (`EncySoftware.CAMAPI.SDK.Net`) that the store can install:

- A prerelease SDK is refused.
- An SDK newer than the store ceiling is refused. The store reports the ceiling as `registrableSdk` at its `/api/sdk` address (see External references); pin the SDK package to that version or lower.
- An SDK newer than any released version of the CAM system gives a warning: the extension needs a version of the CAM system that is not released yet.
- A paid extension must carry the `EncySdk` stamp: the CAM system neither installs nor loads a paid extension without it. SDK 3.0.19 and newer put the stamp into the build. The store looks for it in the `.dll` named after the manifest: `<Name>.settings.json` → `<Name>.dll`. The publication form reports a missing stamp as soon as the package is uploaded.

Reserved functionality

`package.info.json` must contain `reservedDomains`: the Domains of Schedule A the extension works in, or `[]` if none. A package without it is refused on every route (see Publishing).

Versions

- Versions follow NuGet semantic versioning. Prerelease versions are allowed and marked.
- A published version cannot be replaced. To fix a package, increase the version.
- The card shows the highest stable version. An older version number can be published and does not change the card.

Package id and ownership

- The first developer who publishes a package id becomes the owner of the extension. New versions come from the owner, the members of the owning organization or the collaborators (see Organizations and collaborators).
- Ids that differ only in punctuation share the same card address, so the store refuses an id that differs from an existing one only in punctuation.

- If an id is already taken or reserved, choose another one or ask a moderator.

Limits

Limit	Value
Package size	200 MB
Unpacked size	Four times the package size, at least 32 MB and at most 128 MB; up to 20 000 files
Package id	100 characters
Version, name, each package tag, authors, link	255 characters
Description	4000 characters
README	65 536 characters
README translations	20, each 65 536 characters
Picture file name	150 characters
Your own tags on the form	10 tags, 32 characters each
Icon uploaded on the form	PNG or JPEG, at least 32 px on each side
Publications per account	50 per day, 20 attempts per hour

Safety checks

- The store scans each package with an antivirus; an infected package is refused.
- An archive that is damaged, contains paths leading outside the package, or whose table of contents disagrees with its entries is refused.
- The moderator sees hints about native code, process start, network and registry access in the package. These hints are not a verdict, but expect questions if your extension uses them.

Review also checks the rules that no automatic check covers: no code from outside the package, disclosed network use, quality and machine-output requirements (see Store rules and your obligations).

Publishing

Publishing starts with the Publish button in the header of the store website. The page offers two routes as tiles — Publish from GitHub and Upload a .nupkg — and remembers your choice; the AI assistant reaches the same store from your editor. Until you have become a developer the page sends you to do that first (see Getting started).

From a GitHub repository

The template repository contains the workflow that builds the extension; the store packs and publishes it.

1. Create a repository from the template (Use the template on the Publish page). Write your code in `src/`; the card text lives in `src/readme.md` and `src/package.info.json`.
2. Link your GitHub account once: Account settings → Profile → Publishing from GitHub → Sign in with GitHub. After that every repository of yours publishes without tokens. GitHub links whoever is signed in there, so check the account first.
3. In your repository open Actions → publish-to-ency-store → Run workflow and leave the fields empty: the workflow picks the next version by itself. From a terminal: `gh workflow run publish-to-ency-store`.

The result of each run, failures included, appears in My published (see Review, updates and visibility). Bringing an existing project over is described in the template README, section Quick start.

The GitHub route publishes in your own name. To publish for an organization, upload the package (see Organizations and collaborators).

From an AI assistant

The `ency-extension-mcp` server lets an AI assistant create, check and publish an extension from Cursor, Claude Code or another MCP client, with or without a GitHub repository. Its installation, tools and commands are described in the ency-extension-mcp README.

Upload a package

Use this route for code outside GitHub, a one-off release, a package your own build already produces, or a publication for an organization. Every version is uploaded the same way, by hand.

1. Build the package. The packing tool of the CAM system does it in one command:

```
ExtensionManagerCLI.exe pack-library --folder <build output folder> --output
<folder for the package>
```

`ExtensionManagerCLI.exe` is in the `Bin64` folder of the CAM system installation. The folder must contain `package.info.json`; `ExtensionManagerCLI.exe inspect --folder <build output folder>` writes a draft of it. The tool adds the marker tag and leaves out the files that already ship with the CAM system. If you pack with your own tooling, follow Package requirements.

2. On the Publish page choose Upload a .nupkg and drop the file onto the upload area, or click it to pick the file.

3. The form opens with the package id, version and authors, the store's warnings and a preview of the card. Fill it in (see The publication form).

4. Click Submit for review. The page shows Published with a link to the card. If the store refuses the package, the form says why; the refusal is also kept in My published (see Review, updates and visibility).

Nothing is sent until you submit: Choose another package takes you back.

The publication form

The left column is the store card.

Pricing. Free — anyone installs it. Paid — the extension runs only with a licence from the store; a package with `"licensed": true` arrives with Paid already chosen. Paid cannot be undone in later versions. While the extension has no price, Paid also asks for the plans and the trial: they go on the card once the version is approved and reaches the feed. If your paid profile or payouts are not ready, the form shows the step that is left with its button (see Selling and payouts). A paid package must carry the SDK stamp (see Package requirements).

Publish as. Shown only if you belong to an organization: Myself or the organization. A member without a developer registration of their own publishes only for the organization. A new extension belongs to whom you choose; a new version of an existing one goes to its owner (see Organizations and collaborators).

Category. What the extension does, or Let a moderator sort it.

Tags. The package tags stay as they are. Add up to 10 of your own: the store suggests tags it already knows, so that users find your extension under the same words. Your tags stay on later versions until you remove them here.

Icon. Click or drop a PNG or JPEG picture: it is cut to a square and shown round on the card. The uploaded icon stays on later versions; Use the package's picture instead brings back the picture from the package.

Card preview and README show the card approximately as users will see it. The live card changes once a moderator approves the version.

The right column is the reserved functionality declaration, and below both columns are the Declarations (see Reserved functionality). Submit for review stays inactive until both are complete.

Reserved functionality

Some capabilities of the CAM system are licensed separately; Schedule A of the Publishing Policy lists them as Domains. In the Domains open to third parties, an extension may provide such a capability only behind a check that the user holds the licence for that Domain; some Domains are closed altogether (see Store rules and your obligations).

Every submission declares its position, and a submission without the declaration is refused:

- In the package — `reservedDomains` in `package.info.json`, `[]` if the extension provides nothing from Schedule A. The format is described in the template README, section What the store asks you to declare.
- On the form — Does your extension do any of the following? Answer No, or Yes and tick the areas; the table shows the licence each area requires. A new version starts with the previous answer. The form tells you at once if the answer disagrees with the package.
- Declarations — three statements to tick: the extension gives no reserved capability to a user without the matching licence; nobody distributes add-ons or unlocks for it outside the store; the submission is complete and accurate.

The first publication with a given answer is confirmed in the browser. Later versions with the same answer carry it forward, until the answer, Schedule A or the wording of the confirmation changes.

The declaration must match what the extension does: review checks the output, not only the answer, and a materially wrong declaration counts as inaccurate information. A version that adds, removes or moves a capability between Domains needs a new declaration. If you are not sure which Domains apply, ask the store team for a written determination before you build.

Review, updates and visibility

Moderation

A moderator reviews every version before it reaches the store feed: the first version of an extension and each new one. Until the first version is approved, the extension is not in the catalog, in search or on the store feed of the CAM system. To try your build meanwhile, install it from your machine (see Build and test locally).

The moderator checks the card, the declarations and the package against the store rules (see Store rules and your obligations), and sets the category if it is missing. Review aims to finish within 10 business days of a complete submission.

A rejection comes with a reason: the rule relied on, the facts, and what change would make the extension publishable. You see it in My published. To resubmit, fix the extension and publish it again; the same version number is fine, since a rejected version never reached the feed. A rejection can also be appealed (see Store rules and your obligations).

New versions

Increase the version and publish it the same way as the first one. It goes to the moderator too, and the catalog keeps the previous version until the new one is approved. Then users see the update on the Extensions tab of the CAM system.

Versions on their way in My published shows where each version is: Submitted, Approved · signing, Published, Rejected with the reason, or Withdrawn. While a version waits for review, Withdraw takes it back.

Without a new declaration and review, a new version must not:

- add, remove or move a capability between reserved functionality Domains;
- add a capability of a higher licence tier than the extension was published with;
- remove or weaken a licence check.

A published version cannot be replaced or removed: to fix a mistake, publish the next version. Security fixes can be reviewed faster on request.

Keep the extension working with supported releases of the CAM system, and tell the store team at least 60 days before you stop maintaining it.

My published

Account settings → My published lists your extensions — your own, your organization's and those you collaborate on, each group under its own heading — with the version, the date of publication, the state

and the price.

State	Meaning
on moderation	The first version waits for a moderator
rejected	The reason is shown under the row; publish a new version to resubmit
accepted, not in the catalog yet	Accepted, but not yet on the feed
published	In the catalog
hidden	You removed it from the catalog
hidden by a moderator	A moderator took it off the catalog; only a moderator can return it
building…, build failed	The GitHub route; a failed build names the step and links to the log and the run

Version not taken from the feed under a row explains why a new version is missing.

Refused at submission at the top lists the versions the store refused on the spot, before moderation: what the store said, when, and how many attempts. The same mark appears under the row of the extension. Fix what the message says and publish again (see Troubleshooting).

The needs attention filter shows the rows that wait on you: failed builds, refused submissions, rejections and extensions on moderation.

Actions in a row:

- Install in ENCY — for a free extension in the catalog: installs it in your CAM system, starting it if needed.
- Download .nupkg — the package of the latest version, to install from a file or to take to a machine without internet.
- Change the category.
- History — every version of the extension, its steps and refused submissions.
- Collaborators — invite co-authors (see Organizations and collaborators).
- Move into an organization — for your own extension (see Organizations and collaborators).
- Remove from the catalog and Return to the catalog (see Hiding an extension).
- The Price cell opens the price window: plans, trial and free licences for ENCY resellers and the ENCY team (see Selling and payouts).

Everything else on the card — name, description, README, pictures — comes from the package and changes only with a new version. Tags and the icon can also be changed with the next publication (see Package requirements).

Hiding an extension

Remove from the catalog hides the extension from the catalog and search; the card opens by its direct link marked Unlisted. Installed copies keep working, and users keep their licences, but nobody new can buy it or take a trial. Return to the catalog brings it back.

If a moderator hid the extension, only a moderator can bring it back. A harmful extension can be taken down for good: it stops downloading and selling, and every licence issued for it, bought ones included, stops working.

Extensions and versions cannot be deleted from the store. After you leave the store, users keep their licences, and you keep providing them with security and critical-defect fixes for 12 months (see Store rules and your obligations).

Selling and payouts

Selling is optional: most extensions are free. If you want to charge, this page takes you from the first switch to money in your bank account.

Licence check and price

Two separate settings decide what users pay:

- Licence check — the extension runs only with a licence from the store: bought, a trial or given free. It is set by `"licensed": true` in the package or by Paid on the publication form, and it stays on every later version.
- Price — the plans users can buy and the length of the trial. You set them on the publication form of a paid version or later in My published.

Licence check	Price	What users get
Off	—	A free extension: anyone installs it
On	Not set	Nothing yet: no trial and no purchase; the card says Price not set
On	Free	Anyone takes a free perpetual licence with one click
On	Plans	A trial, then a purchase or a subscription

Setting a price turns the licence check on for every later version. Users who installed a version while it was free keep using it.

A free extension must be free in substance (see Store rules and your obligations).

Get ready to sell

1. Paid profile. Choose Free and paid extensions when you become a developer, or click Enable paid extensions later: on the publication form or in Account settings → Developer. The paid profile is open to companies and to individuals acting for their trade, business or profession.
2. Payouts. Click Set up payouts on the publication form or in Account settings → Earnings, and enter your bank and identity details on the Stripe form; the store never sees them. If Stripe needs more,

Continue setup at Stripe takes you back. Payouts are open to companies incorporated, and individuals living, in the United States, the United Kingdom, the EEA, Canada or Switzerland.
3. Approval. Submit a paid version with your payouts ready: the moderator's approval enables paid extensions for you, and the extension comes out with its price. If you price an extension that is already published, the store team enables paid extensions once your payouts are ready.

The publication form and the price window walk you through the same steps: if something is missing, they say what and offer the button for the next step.

Extensions of an organization cannot be sold yet: payouts are made to people, not to organizations.

Set a price

With a paid version. On the publication form choose Paid under Pricing, tick the plans and enter the trial (see below). The price goes on the card the moment the version is approved and reaches the feed. The form asks for plans only while the extension has no price; after that the price stays as it is and changes only in My published.

For a published extension.

1. In My published, click the Price cell of the extension.
2. Choose Paid, tick the plans and enter the trial.
3. Click Save price.

The plans:

- One-time — a perpetual licence, from $19;
- Monthly — a subscription, from $10 a month;
- Yearly follows from the monthly price: ten monthly payments, two months free.

Trial — how many days users try the extension before they buy: from 14 to 30. Leave it empty to use the store's default, which the field shows. A user gets the trial once.

Prices are in US dollars, up to $10,000. Users see every plan you tick and choose one. Only the owner sets prices; members of an organization and collaborators see them but cannot change them.

Changing a price. New buyers pay the new price. Running subscriptions keep the price they were bought at, and perpetual licences stay as they are. A plan you untick disappears for new buyers; subscriptions bought on it keep renewing.

Free. Choose Free to give the extension away. With the licence check on, anyone gets a free perpetual licence on Get it free, and keeps it if you set a price later.

Free for ENCY resellers and Free for the ENCY team. With the licence check on, these ticks in the price window give the chosen group a licence for a year, renewed while they stay in the group. It is not a sale:

nothing accrues to you. Untick to stop new ones; licences already given run out their year.

Your own licence. When a paid version of your extension reaches the feed, you get a perpetual licence for it. Members of the organization and collaborators take theirs with Get my license on the card.

Your share

- The store keeps a 30% commission on each sale and renewal; 70% of the price accrues to you.
- The store sells to users and charges them; taxes on what you receive are yours.
- Refunds are made by the store team (see What your users see). A refund or a lost card dispute takes your share of that sale back.

The Commercial Terms govern prices, commission, payouts, refunds and taxes (see External references).

Payouts

Account settings → Earnings shows your balance, your payouts and the history of your sales and renewals.

- Monthly payout. When the balance reaches $100, it goes out in the monthly payout at the start of the next month, after the store team confirms the batch. A balance under $100 is paid out at the turn of the year.
- First payout. No earlier than 60 days after your first sale.
- Early payout. Pay me now sends the balance before the monthly payout, for a fee of $5 plus 1% of the amount. It needs at least $50 on the balance and waits for the store team.
- Checks. Every payout passes a sanctions check before it leaves.
- To your bank. The store transfers the money to your Stripe account; Stripe pays it to your bank on its own schedule. Open your Stripe dashboard shows the transfers.

A payout that fails or is held back stays on your balance, and Earnings says why.

What your users see

This page follows a user from the catalog to your extension running in the CAM system, so that you know what your customers experience and what to tell them.

In the catalog

Users browse the store website or the Extensions tab of the CAM system and filter by Free and Paid. The button on a card says what awaits them:

- Install — a free extension;
- the lowest price, for example $10/month — an extension on sale;
- Price not set — an extension with a licence check whose author has not priced it yet: nobody can take a trial or buy it until then;
- View — a licensed extension given away free.

The card itself shows every plan, the licence of the CAM system a gated extension requires, and the minimum version of the CAM system. A version newer than the user's CAM system is a warning, not a block.

Trial

A signed-in user takes a trial with Get N-day trial · then $X: the length is the one you set, from 14 to 30 days, and the button names the price that follows. No card is needed, and nothing is charged when the trial ends. A trial is issued once per account and extension, and only while the extension is in the catalog and has a price. The card then shows when the trial ends and offers Install in ENCY; the Buy button stays. When the trial is over, the card says so and the user buys.

Free for ENCY resellers and the ENCY team

If you ticked these groups in the price window, their members see Get it free as an ENCY reseller or Get it free as ENCY team instead of the trial. They get a licence for a year, renewed when they sign in while they stay in the group.

Purchase

1. The user picks a plan — One-time, Monthly or Yearly — and clicks Buy or Subscribe, before the trial or after it.
2. Payment goes through Stripe Checkout, by card, in US dollars; the store never sees the card.
3. Back in the store, the card says You own this extension and offers Install in ENCY.

Each purchase is one licence for the buyer's own account.

A yearly plan is sold only to users whose licence of the CAM system covers the whole year ahead; others choose monthly or one-time. Nobody can buy the same extension twice or hold two subscriptions to it.

If the payment went through but the licence has not arrived yet, the card says the licence is on its way and updates by itself.

Installation and updates

Install in ENCY opens the CAM system on the user's machine, starting it if needed, and installs the extension. Download .nupkg gives the package file, to install from the Extensions tab or to take to a machine without internet.

The licence belongs to the user's account: the CAM system must be signed in to the same account. Once the licence is on the machine, the extension works without a connection to the store; renewals arrive online.

A purchase covers every later version: updates appear on the Extensions tab and install without paying again.

Subscriptions

A subscription renews every month or year from the day of purchase.

- A failed payment. The subscription shows Payment failed, and the extension keeps working while Stripe retries, for up to 22 days. A successful retry restores everything; if all retries fail, the subscription ends and the licence with it.
- Cancellation. Manage billing in My subscriptions opens the Stripe portal, where the user changes the card, downloads invoices or cancels. A cancelled subscription works until the end of the paid period.

Refunds

Users ask the store team for a refund; there is no refund button. A full refund ends the purchase: the licence is revoked, and a subscription stops at once. A partial refund leaves the licence in place. Either way, your share of the refunded amount comes off your balance (see Selling and payouts).

The user's account

Section	What it shows
My licenses	Every licence, trial or full, with its state and end date
My payments	Purchases and every charge: first payment, renewals, refunds

Section	What it shows
My subscriptions	Plan, amount paid so far, next payment, Manage billing

When you change something

You…	Users
Change the price	New buyers pay the new price; running subscriptions keep theirs
Change the trial	New trials get the new length
Untick a plan	It disappears for new buyers; subscriptions bought on it keep renewing
Make it Free	Anyone can take a free perpetual licence
Remove it from the catalog	Installed copies and licences keep working; nobody new can buy it or take a trial
Publish a new version	Owners of a licence update it free of charge

Organizations and collaborators

Two ways to publish together:

- An organization publishes for a company. The company accepts the terms once, and its members publish on its behalf.
- Collaborators help with one of your own extensions. Each of them is a developer in their own right.

Create an organization

Your company. If your account of the CAM system belongs to a company and at least one colleague from it also uses the store, Account settings → Organizations offers it under Your company. Click Bring it in: you become the owner, and everybody whose account is in this company is a member without an invitation.

Another company. Click New organization, enter the company name as the store should show it, and click Create.

Then, in both cases:

1. Click Fill in the company details. The wizard is the one of Becoming a developer, without the Type step: you fill in your details as the company representative and the company details.
2. Accept the terms on behalf of the company with I Agree. You confirm that you may bind it.
3. Invite the people who publish with you and are not members by company: enter the email of each in the people table and click Invite. A person can be invited once they have signed in to the store at least once.

On the card the organization is named by its public developer name. An organization that has not accepted the terms and has no extensions can be deleted with Delete.

Join an organization

If your account belongs to the company, you are a member as soon as the organization is in the store, without an invitation, and you stop being one when your account leaves the company.

Anybody else is invited. An invitation appears in Account settings → Organizations → Invitations; the store sends no email. Accept makes you a member, Decline refuses. Leave takes you out.

Members do not sign the organization's agreement and need no developer registration of their own: the owner's acceptance covers what they publish for it.

Publish for an organization

Upload the package and choose the organization under Publish as on the form (see Publishing). A new extension then belongs to the organization; a new version of an existing extension always goes to its

owner. The GitHub route publishes in your own name.

Move your extension into an organization

In My published, click Move into an organization in the row of your own extension, choose the organization and click Move. Its members then publish its versions, and the owner of the organization manages it. Your collaborators leave the extension.

- An extension with prices or sales cannot be moved.
- A version on its way to the store must be published or withdrawn first.
- The organization must have accepted the terms.

You cannot take the extension back yourself. Only the owner of the organization can take it out and hand it to a member, who must have accepted the developer terms as an individual.

The organization's extensions appear in My published of every member under the heading of the organization.

	Owner	Member
Publish new versions, hide and show, change the category	Yes	Yes
Company details, terms, invitations, removing people	Yes	No
Prices	Yes	No

Organizations cannot sell yet: payouts are made to people (see Selling and payouts).

Hand over the organization

The owner clicks Make owner in the row of a member. The member finds the offer under Invitations and clicks Accept; until then the organization stays with the owner, who can Withdraw the offer. The previous owner becomes a member.

Collaborators

Invite a co-author to one of your own extensions: My published → Collaborators in its row → enter their email → Invite.

A collaborator must be a developer: they accept the terms as individuals first, in Account settings → Developer. They find the invitation in Account settings → Organizations and click Accept. Then they

publish new versions of the extension and hide or show it; prices, money and the list of collaborators stay yours. Remove in the same window, or Leave on their side, ends the collaboration.

Collaborators publish by uploading the package: the GitHub route publishes only extensions of your own.

Troubleshooting

Store messages

The store refuses a package with a message that names the problem. The most common ones:

Message begins with	What to do
The ⋯ Extension Store Developer Agreement (vX) has not been accepted	Become a developer and click I Agree (see Getting started), then publish again.
The ⋯ Extension Store terms changed for the account ⋯	Open Account settings → Developer, accept the new versions with I Agree, then publish again.
The store requires a declaration	Add `reservedDomains` to `package.info.json`, `[]` if the extension provides nothing from Schedule A (see Publishing).
The package's manifest has no "reservedDomains"	The form answer needs the key in the package too: add `"reservedDomains": []` to `package.info.json`.
The package has no 'ency-extension' tag	Add the marker tag to `tags` in `package.info.json`, or pack with the template or `ExtensionManagerCLI` (see Package requirements).
The .nupkg has no *.settings.json manifest or no .dll inside	Check the package layout (see Package requirements).
The manifest has no "module_path"	Add `module_path` to the manifest (see Extension anatomy).
Manifest has no 'extensions' array / Manifest declares no extension points at all	Declare at least one entry point in `extensions`.
The package says two different things about itself	Make the id and version in `.nuspec` and `package.info.json` match, and rebuild.
Built against SDK x, which cannot be installed yet	Pin the SDK package to the version the message names, rebuild and publish again (see Package requirements).

Message begins with	What to do
Version X of Y is already on the feed / This version is already published	Increase the version and publish again.
You are not the owner of extension X	The package id belongs to another developer. Choose another id, or ask its owner to add you as a collaborator.
The name X would share its address with the existing extension Y	Choose an id that differs from Y in more than punctuation.
The name X is reserved by another publisher	Choose another id, or ask a moderator to release the name.
X is in the store as an organization	Ask the owner of the organization to invite you, and publish on its behalf (see Organizations and collaborators).
X has not accepted the Developer Agreement yet	The owner of the organization accepts the terms under Account settings → Organizations.
Your developer profile is Free, and a paid extension needs the Paid profile / Your developer profile is for free extensions only	Publish without Paid, or click Enable paid extensions (see Selling and payouts).
A paid extension needs your payout account with Stripe	Click Set up payouts on the form or in Earnings, finish the Stripe form and submit again.
… carries no EncySdk stamp / A paid extension needs its dll named after the manifest	Rebuild against SDK 3.0.19 or newer, with the `.dll` named after the manifest (see Package requirements).
The card takes up to 10 tags of yours / The tag … is N characters long	Keep at most 10 tags of your own, 32 characters each.
The card icon …	Upload a PNG or JPEG picture of at least 32 px on each side.
… is N characters long; the store keeps at most M	Shorten the field (see Package requirements).

Message begins with	What to do
This store accepts N publishes a day from one account / Too many publish attempts	Wait and try again later.
The package was refused by the virus scanner	Remove the flagged file from the package. If you believe the finding is false, contact the store team.
The virus scanner is not answering / The licensing service did not answer	Try again in a few minutes; the package was not published.
Payouts cannot be set up right now: Stripe is waiting for a confirmation from the store	The store team has been told; try again later.
… has prices or sales: a paid extension cannot change hands	An extension with prices or sales stays with its owner and cannot be moved into an organization.

Refused packages stay listed under Refused at submission in My published (see Review, updates and visibility).

In the CAM system

The extension is installed but not visible. Check on the Manage page of the Extensions tab that the library and its entry point are enabled, and look for it where its entry point type puts it (see Extension anatomy). Restart the CAM system if it still does not appear.

Installing a new build fails with an access error. The previous build is still loaded. Close the CAM system and install again (see Build and test locally).

A published extension is not in the catalog. Each version appears after the moderator approves it. Check its state under Versions on their way in My published (see Review, updates and visibility).

The card has no icon. Upload one on the publication form, or put PNG or JPG pictures into `screenshots/` of the package; the first one becomes the icon.

Users cannot take a trial of a paid extension. The extension has no price yet: the card says Price not set. Set the price (see Selling and payouts).

Not sure it is allowed

- Whether a capability needs a licence check, or may be published at all — ask the store team for a written determination before you build (see Store rules and your obligations).
- Whether a name is acceptable — ask the store team before you publish.
- How to give an extension to one customer or an internal team only — ask for a private listing.

See Contacts for the address.

External references

Store

Reference	URL
ENCY Extension Store	https://apps.encycam.com/
Publish page	https://apps.encycam.com/publish
Your account: profile and GitHub link	https://apps.encycam.com/account
Developer: registration, terms, paid extensions	https://apps.encycam.com/account/developer
My published	https://apps.encycam.com/account/published
Organizations, invitations and collaborations	https://apps.encycam.com/account/organizations
Earnings: payouts and sales	https://apps.encycam.com/account/earnings
SDK ceiling of the store (`registrableSdk`)	https://apps.encycam.com/api/sdk
Store feed for ENCY	https://apps.encycam.com/api/nuget/v3/index.json
Extension Store in the ENCY user manual	https://docs.encycam.com/ENCY/3/en/162964210.html

Development

Reference	URL
Extension template: project, build, publishing from GitHub	https://github.com/EncySoftware/ency-extension-template
MCP server for AI assistants	https://github.com/EncySoftware/ency-extension-mcp
API usage examples and their documentation	https://github.com/EncySoftware/cam-api-examples/tree/v3/main/docs
Design rules for extensions	https://github.com/EncySoftware/cam-api-examples/blob/v3/main/docs/general/design-

Reference	URL
	rules.md
CAM API reference	https://docs.encycam.com/technical/CAMAPI/3/en/
CAM API lessons	https://docs.encycam.com/technical/CAMAPI/3/en/src/Lessons/Main.html

Store terms

Reference	URL
All documents of the ENCY Extension Store	https://encycam.com/legal/extension-store/
Developer Agreement	https://encycam.com/legal/extension-store/developer-agreement/
Publishing Policy, with Schedule C — Enforcement and Appeals	https://encycam.com/legal/extension-store/publishing-policy/
Schedule A — Reserved Functionality	https://encycam.com/legal/extension-store/reserved-functionality/
Schedule B — Technical Requirements	https://encycam.com/legal/extension-store/technical-requirements/
Commercial Terms	https://encycam.com/legal/extension-store/commercial-terms/
SDK Licence Terms	https://encycam.com/legal/extension-store/sdk-licence/
Trade Mark Guidelines	https://encycam.com/legal/extension-store/trade-marks/
Standard Extension EULA — applies when you have no end-user terms of your own	https://encycam.com/legal/extension-store/standard-eula/
Changes to the documents	https://encycam.com/legal/extension-store/changelog/
Reserved functionality areas and their licences, for tools	https://apps.encycam.com/api/legal/reserved-functionality

Contacts

Purpose	Address
Questions, written determinations on reserved functionality, naming questions, private listings, complaints	developers@encycam.com
Vulnerabilities, data breaches and safety-relevant defects — within 72 hours	security@encycam.com
Appeals against decisions — within 30 days, with the reference number from the notice	appeals@encycam.com

Related resources

Reference	URL
ENCY portal	https://encycam.com/
ENCY technical documentation	https://docs.encycam.com/portal/