Getting Started

What is ShortcutXL?

ShortcutXL is a general-purpose AI agent with Excel superpowers. It's designed for power users — analysts, modelers, operators — with the goal of 5-10x'ing your productivity.

ShortcutXL has deep Excel capabilities that Excel plugin environments do not provide — VBA, Power Query, iterative recalcs, sensitivity tables, etc — and can work across multiple workbooks simultaneously. It has full access to your local filesystem, and integrates with email, APIs, databases, and financial data sources.

ShortcutXL is a terminal application that runs alongside Excel, not inside it. This is intentional — it's designed to be easily parallelized. You can run multiple instances working on the same workbook or across different workbooks, or have one agent build a model while another pulls data.

Quick Install

1. **Open a terminal** — Press `Win + R`, type `cmd`, press Enter.
2. **Install Node.js** — `winget install OpenJS.NodeJS.LTS --source winget` (then close and reopen your terminal)
3. **Install ShortcutXL** — `npm install -g shortcutxl`
4. **Launch** — `shortcut`

On first launch, ShortcutXL automatically sets up your environment. A **preflight** script installs prerequisites (Git, Python packages, Chrome) and registers the Excel add-in. If anything needs manual attention, an **installation agent** takes over and walks you through it conversationally. The whole process takes a few minutes — just follow the prompts.

After setup, just run `shortcut` to start. See Installation for detailed troubleshooting and system requirements.

Commands & Features

Authentication

`/login` — Same Shortcut login you've always used. Signs you into your Shortcut account so the agent can access AI models on your behalf. No separate API keys needed.

`/logout` — Signs you out of your Shortcut account.

Skills

`/download-skills` — Downloads your pre-existing skills from Shortcut's cloud to your local machine. If your team has published skills or you've created skills on another machine, this syncs them locally so the agent can use them.

`/upload-skills` — Uploads your local user skills from this machine to Shortcut's cloud. New skills upload directly; if a cloud copy already exists and differs, Shortcut asks for approval before replacing it. Team skills are read-only here and are never uploaded.

`/skills` — Shows all currently loaded skills, so you can see what the agent has access to and verify your skills downloaded correctly.

Modes

ShortcutXL has three modes that control how the agent interacts with your spreadsheet. Switch between them at any time.

`/action` – The default working mode. The agent reads and writes to your spreadsheet – building models, writing formulas, formatting cells, creating charts. This is where the work gets done.

`/plan` – Read-only planning mode. The agent explores your workbook structure, asks clarifying questions, and produces a structured implementation plan before touching anything. Great for complex or ambiguous tasks like "build me a DCF" where you want to align on the approach first. Once you approve the plan, it switches to Action mode to execute.

`/ask` – Read-only analysis mode. The agent reviews, audits, and explains your spreadsheet without making any changes. Use this when you want to understand someone else's model, check formulas for errors, trace dependencies, or get a second opinion on calculations.

Models

`/model` – Switches between available AI models:

- **Default model** – Best for most tasks: building models, complex formatting, and multi-step workflows.
- **Reasoning/auditing models** – Useful when you want a second pass on formulas, logic, or numerical checks.
- **Fast models** – Useful for simple tasks where speed and cost matter more than maximum capability.

Thinking Level

`/thinking` – Controls how much the AI "thinks" before responding. Higher levels mean more reasoning but slower responses.

- **Low** – Recommended for most work. Best balance of speed and intelligence.
- **Medium / High** – Great for complex logical tasks where you want the AI to reason deeply – intricate formula chains, multi-sheet model builds, or tasks you want to kick off and step away from.
- **Minimal / Off** – Fastest responses. Fine for trivial tasks like "make this cell bold."

Approval Mode

By default, ShortcutXL asks for your approval before every edit. Each time the agent modifies cells, you see a diff showing exactly what changed – before and after values for every cell – and you can Accept, Reject, or Accept All.

`Shift+Tab` – Toggles between "ask before edits" and "approve all edits" for the current session. When you're in a flow and trust what the agent is doing, flip to auto-approve so it can work without interruption. Flip back any time you want to review changes again.

This can also be toggled from `/settings` → Auto-approve edits.

Permissions

ShortcutXL keeps runtime permissioning separate from spreadsheet approval.

`/permissions` – Shows the effective runtime permission boundary and lets you manage:

- **Approved workspaces** – Broad folders the agent can trust by default
- **Approved files & folders outside workspaces** – Narrow exceptions outside those workspaces
- **Spreadsheet auto-approve default** – Whether new sessions ask before spreadsheet changes
- **Skip runtime permissions** – Explicitly bypass runtime filesystem and shell permission checks

Sandbox

The sandbox provides explicit permission controls for what the agent can access on your machine – file read/write paths and network access. It runs shell commands inside an isolated Linux environment via WSL2, so the agent can only touch what you've allowed.

`/sandbox` – Opens the sandbox settings menu where you can customize your permissions:

- **Enable / Disable** – Turn the sandbox on or off

- **Network mode** – `none` (no network, the default), `proxy` (you approve each domain the agent wants to reach), or `full` (unrestricted)
- **Read paths** – Which directories on your machine the agent can read from
- **Write paths** – Which directories the agent can write to
- **Allowed domains** – Pre-approved domains for proxy mode (e.g., `*.github.com`, `*.pypi.org`)

Settings

`/settings` – Opens a menu for general preferences:

- **Auto-approve edits** – Whether the agent asks before each edit or applies changes automatically
- **Theme** – Color theme for the terminal interface
- **Quiet startup** – Skip the verbose startup output for a cleaner launch
- **Collapse changelog** – Show a condensed changelog when ShortcutXL updates
- **Upload conversation data** – Controls whether session traces are uploaded for observability and debugging
- **Show images** – Render images inline in the terminal (if your terminal supports it)

Session Management

`/resume` – Pick up where you left off. Shows a list of your previous sessions so you can continue an earlier conversation with full context preserved.

`/clear` – Starts a fresh session, clearing the current conversation history.

`/quit` – Exits ShortcutXL.

Learn More

- Installation – Detailed setup guide, preflight steps, troubleshooting
- Core Features – Excel capabilities, general agent, modes, approval flow, parallel agents
- Data Integrations – Bloomberg, FactSet, Google, Outlook, databases, and 30+ connectors
- MCP Servers – Connect external tools via the Model Context Protocol (GitHub, Slack, and more)
- Document Creation – PDFs, PowerPoints, screenshots, and SEC filings
- Extensibility – Skills, extensions, and the skill creator
- Permissions – Runtime permissioning, approved workspaces, external path grants, and bypass mode
- Security & Privacy – Sandbox details, credential management, and execution sandboxing
- CLI Reference – All flags, options, environment variables, and package management
- Claude Code & Codex Integration – Use ShortcutXL as a plugin from Claude Code or Codex

Installation

Step 1 — Open a Terminal

Press `Win + R`, type `cmd`, and press Enter. Or search for "Command Prompt" or "PowerShell" in the Start menu.

Step 2 — Install Node.js

ShortcutXL requires Node.js 18+. If you already have it, skip to Step 3.

```
winget install OpenJS.NodeJS.LTS --source winget
```

After installing, close and reopen your terminal, then verify:

```
node --version
```

Step 3 — Install ShortcutXL

```
npm install -g shortcutxl
```

Step 4 — Launch

```
shortcut
```

On first launch, ShortcutXL runs a guided setup (see below). After that, just run `shortcut` to start.

First Launch — Preflight

On first run, ShortcutXL runs a preflight setup that prepares your environment. This is automatic — you just follow the prompts.

What preflight does

On Windows:

1. **Git** — Checks for Git (needed for shell operations). If missing, offers to install it via `winget`. You may see a UAC prompt — click Yes.
2. **Python packages** — Installs `pywin32` and `playwright` into ShortcutXL's embedded Python runtime. No system Python needed — Python 3.12 ships inside the npm package.
3. **Chrome** — Checks for Chrome (used for SEC filing PDF conversion). If missing, offers to install it via Playwright.
4. **Excel add-in** — Registers ShortcutXL with Excel so it loads automatically on startup. This writes a small user-level registry key (no admin required).
5. **Excel connection** — Asks you to open Excel (any workbook), then verifies the add-in loaded and the HTTP connection is working.

6. **Smoke test** — Runs a round-trip test: agent → HTTP → add-in → Python → Excel COM → back. If this passes, setup is complete.

If something fails

Preflight is best-effort. If any step fails, ShortcutXL's installation agent takes over automatically. It reviews what preflight accomplished, focuses on what's left, and walks you through fixing it conversationally. You don't need to do anything special — just follow the agent's instructions.

After Setup

Once preflight (or the installation agent) completes successfully, you're done. From now on, just run:

```
shortcut
```

Excel automatically loads ShortcutXL on startup. No manual setup needed again.

What to try first

- **Ask it to do something in Excel** — "Build a DCF model", "Format this table", "Pull data from Sheet2 into a summary"
- **Explore capabilities** — Ask the agent what it can do, or try `/skills` to see loaded skills
- **Connect data sources** — Run `/connect` to set up database or API connections
- **Create custom skills** — Tell the agent "help me create a skill for [your workflow]" and it will build one for you

Updating

```
npm update -g shortcutxl
```

ShortcutXL tells you when a new version is available. The Excel add-in auto-syncs with the agent — no separate update needed.

Uninstalling

```
npm uninstall -g shortcutxl
```

This removes the CLI and all shipped files. The Excel registry key is cleaned up automatically on next Excel launch when the add-in file is no longer found.

Core Features

Excel Capabilities

ShortcutXL runs Python inside the Excel process via a C add-in, giving it full access to the COM object model — the same interface that VBA uses. Everything the Shortcut web plugin can do, plus everything that only COM/VBA can access:

VBA macros, Power Query, Power Pivot / Data Model (DAX measures, relationships), Goal Seek, Solver, Scenario Manager, sensitivity tables, iterative calculation with circular references, 73+ chart types, pivot tables with calculated fields and 70+ filter types, slicers and timelines, conditional formatting (color scales, data bars, icon sets), data validation, structured tables, SmartArt, shapes, sparklines, named ranges, AutoFilter, Advanced Filter, subtotals, row/column outlining, sheet/workbook/range protection, page setup and printing, export to 50+ file formats, freeze panes, multi-workbook with cross-book references, threaded comments, and the full Python ecosystem on top.

General AgentController

Outside of Excel, ShortcutXL is a general-purpose agent running on your machine — similar to Claude Code. It has a full bash shell, access to your local filesystem, and the ability to install and run any tool or library.

- **File system** — read, write, search, and organize files anywhere on your machine
- **Shell** — run any command, script, or program. Install packages, run Python scripts, use git, curl APIs, pipe data between tools.
- **Web** — search the web, fetch URLs, scrape pages, download files
- **Document processing** — read PDFs, images, CSVs, JSON, XML, and 30+ other formats. Extract tables from PDFs with camelot and PyMuPDF. Analyze images and charts with multimodal AI.
- **Code execution** — write and run Python, Node.js, or any language available on your system
- **Office automation** — create PowerPoint presentations (python-pptx), generate PDFs (fpdf2, ReportLab, HTML-to-PDF), read and write Word documents
- **Email & messaging** — send and read emails through Outlook or Gmail, post to Slack

This means ShortcutXL can handle end-to-end workflows that go beyond the spreadsheet — pull data from an API, process it in Python, write it to Excel, format it, generate a PDF report, and email it out, all in one conversation.

Modes

Action Mode (`/action`)

The default. The agent reads and writes to your spreadsheet — building models, writing formulas, formatting cells, creating charts, running macros. Every modification goes through the approval flow (unless you've toggled auto-approve).

Plan Mode (`/plan`)

Read-only. The agent explores your workbook structure, asks clarifying questions one at a time, and produces a structured implementation plan. Great for complex or ambiguous tasks — "build me a DCF", "restructure this model", "this spreadsheet is broken" — where you want to align on the approach before anything gets changed. Once you approve the plan, it switches to Action to execute.

Ask Mode (`/ask`)

Read-only analysis. The agent reviews, audits, and explains your spreadsheet without making any changes. Use it to understand someone else's model, check formulas for errors, trace dependency

chains, verify calculations, or get a second opinion. The agent cannot switch out of Ask mode on its own — it's a safe space for pure analysis.

Review & Approval

Every modification the agent makes to your spreadsheet is shown to you as a diff before it takes effect:

ADDRESS	BEFORE	AFTER	DETAIL
B2	100	=SUM(B3:B10)	formula
C5		$1,500.00	hardcoded

Changes are categorized — formulas, hardcoded numbers, potential errors, large percentages — so you can quickly spot anything unexpected. You can:

- **Accept** — apply the changes
- **Accept All** — auto-approve for the rest of the session
- **Reject** — revert everything to its original state

Read-only operations are auto-approved and never prompt. Shell commands go through a separate approval layer — known-safe commands (ls, cat, grep) are auto-approved, while unknown commands require confirmation.

Toggle between ask-before-edits and auto-approve with **Shift+Tab** at any time.

Parallel Agents

ShortcutXL can spawn multiple subagents that work concurrently. This is how it handles complex, multi-part tasks efficiently — instead of doing things one at a time, it fans out.

- **Document Reader** — specialized for extracting structured data from PDFs and images. Uses programmatic tools (camelot, PyMuPDF, pdfplumber) for large tables and multimodal AI (Gemini Flash, Claude Haiku) for visual content. Cross-validates using both methods for critical accuracy.
- **General** — handles research, web lookups, data processing, batch classification, and spreadsheet operations. Can read and write to Excel.
- **Clone** — a full copy of the main agent that can itself spawn Document Reader and General subagents. Used for truly parallel workstreams.

All subagents run concurrently with a 15-minute timeout each. The agent shows live progress tracking as subagents work.

Use Cases

- Extract data from 5 PDFs simultaneously, then consolidate into one sheet
- Have one agent build the revenue model while another builds the cost model
- Run a data pull and a formatting task at the same time
- Batch-classify hundreds of rows using parallel LLM calls

Session Management

Conversations are persistent. Every session is saved as a JSONL transcript with full branching support.

- `/resume` — pick up any previous session with full context preserved
- `/tree` — navigate your conversation history and branch from any point, like git for conversations

- **Context compaction** — when the conversation gets long, ShortcutXL automatically summarizes earlier context so you never hit a wall. You can also trigger this manually with `/compact`.

Permissions

ShortcutXL has two separate security layers:

- **Runtime permissions** control what the agent can do through its built-in file and shell tools.
- **Sandbox settings** control whether bash runs inside an isolated WSL2 environment with separate filesystem and network rules.

This page is about the runtime permission boundary.

Runtime Permissions

Use `/permissions` to inspect and change the current runtime permission settings.

You can:

- Review whether runtime permissions are enforced or bypassed
- Add or remove approved workspace roots
- Add or remove approved files and folders outside those workspaces
- Change the default spreadsheet approval behavior
- Open the underlying permissions JSON

Approved Workspaces

Approved workspaces are the broad folders ShortcutXL can trust by default.

Inside an approved workspace:

- File tools can read files without extra approval
- In Action mode, file tools can write files without extra approval

Outside approved workspaces, ShortcutXL falls back to narrower approvals or denies the action.

Approved Files And Folders Outside Workspaces

If you do not want to approve an entire workspace, you can grant narrower exceptions:

- A single file
- A single folder
- Read-only access
- Read and write access

These exceptions are useful for shared data folders, exports, or one-off templates that sit outside your main project directories.

Spreadsheet Approval Default

Spreadsheet approvals are controlled separately from filesystem and shell permissions.

Use `/permissions spreadsheet-auto-approve` to choose whether new sessions should:

- Ask before spreadsheet changes, or
- Auto-approve spreadsheet changes

This only affects spreadsheet mutations. It does not disable runtime filesystem or shell checks.

For non-interactive runs, you can also allow headless spreadsheet auto-approval for a single session:

```
shortcut --skip-spreadsheet-permissions -p "..."
```

This widens the headless spreadsheet policy and starts that session with spreadsheet auto-approve enabled. Runtime filesystem and shell permissions are unchanged.

Skip Runtime Permissions

Use `/permissions skip-runtime-permissions` to bypass runtime-owned filesystem and shell permission checks for the current settings profile.

This is intentionally explicit because it widens the agent's access:

- File tools are no longer constrained by approved workspaces or external path grants
- Shell execution no longer waits on runtime permission checks
- Spreadsheet approvals are unchanged

For non-interactive runs, the matching CLI flag is:

```
shortcut --skip-runtime-permissions -p "..."
```

Only use this when you intentionally want the runtime boundary disabled.

Sandbox vs Permissions

Use `/permissions` when you want to control ShortcutXL's built-in runtime boundary for file tools and shell execution.

Use `/sandbox` when you want bash itself to run inside the isolated WSL2 sandbox with its own filesystem mounts and network policy.

These are related but different:

- `/permissions` changes what the runtime allows
- `/sandbox` changes where bash runs and what that isolated environment can access

Document Creation

ShortcutXL can generate PDFs, PowerPoint presentations, and images directly from your spreadsheet data or from scratch.

PDF Generation

Three approaches depending on your needs:

Programmatic (fpdf2, ReportLab)

Best for structured reports with precise layout control.

> *"Create a PDF report of the Q3 financials from Sheet1"*

The AI writes Python code that constructs the PDF page by page — text, tables, images, headers, footers. Builds incrementally: a few pages at a time, verifying as it goes.

HTML-to-PDF (Playwright)

Best for complex layouts, styled content, and web-like formatting.

> *"Convert the dashboard summary into a nicely styled PDF"*

Renders HTML with full CSS support in a headless browser, then exports as a PDF.

Office Conversion (LibreOffice)

Best for converting existing Office files.

> *"Convert this PowerPoint to PDF"*

Uses LibreOffice headless mode to convert `.pptx`, `.docx`, or other Office formats to PDF.

PowerPoint Creation

Creates `.pptx` presentations using the python-pptx library.

> *"Build a 10-slide pitch deck from the data in my model"*

> *"Create a presentation summarizing each sheet as a separate slide"*

Capabilities

- Multiple slide layouts (title, content, section header, blank)
- Text formatting — fonts, sizes, colors, bold, italic
- Tables, images, and shapes on slides
- Theme colors and consistent branding
- Incremental build — slide by slide with verification

Output is saved to your workspace and can be converted to PDF for distribution.

Excel Screenshots

Export any range from Excel as an image.

> *"Screenshot the summary table on Sheet1 and save it as a PNG"*

How it works

1. The AI copies the range as a bitmap to the clipboard
2. Saves the clipboard image to disk using Pillow

Supported formats

- PNG, JPEG (with quality control), BMP, TIFF

Useful for embedding Excel visuals in emails, presentations, or reports.

Document Reading

The AI can extract structured data from PDFs, images, and other documents with high accuracy.

> *"Extract the financial data from this PDF and put it in Excel"*

> *"Read the table on page 3 of this report"*

Extraction methods

- **Programmatic** (camelot, PyMuPDF, pdfplumber) — fast and deterministic for structured tables
- **Multimodal AI** (Gemini Flash, Claude Haiku) — handles visual content, charts, scanned documents, and complex layouts
- **Cross-validation** — both methods used together for critical accuracy

Supported inputs

- PDFs, PNG, JPEG, GIF, WEBP, HTML, CSV, JSON, XML, and 30+ text formats

SEC Filing Retrieval

Pull SEC filings for any US public company and convert them to PDF for analysis.

> *"Get Apple's latest 10-K and extract the revenue breakdown"*

Supported filing types

- 10-K (annual), 10-Q (quarterly), 8-K (current events), DEF 14A (proxy), and amendments

Workflow

1. Looks up the company on SEC EDGAR
2. Fetches filing metadata and identifies the latest filing
3. Converts the HTML filing to PDF using Chrome
4. Analyzes the PDF with multimodal AI to extract the data you need

Files are saved to `~/Desktop/sec/{TICKER}/{form}_{date}.pdf` for future reference.

Data Integrations

ShortcutXL ships with skills that teach the agent how to work with your existing data providers, productivity tools, databases, and business platforms. The agent knows the formula syntax, API patterns, and connection details — you just ask for what you need.

Financial Data Providers

ShortcutXL has skills for all major market data providers and accounting systems. The agent writes provider-specific Excel formulas directly into your cells — data flows through each provider's own Excel add-in, so there's nothing extra to install.

Market Data & Fundamentals — Bloomberg, FactSet, S&P Capital IQ, Refinitiv (LSEG), Morningstar, QuickFS, Calcbench, Nasdaq Data Link, YCharts

Alternative & Private Markets — PitchBook, Preqin, AlphaSense/Tegus, Visible Alpha

Credit & Fixed Income — Moody's Analytics, ICE Fixed Income

Accounting & ERP — QuickBooks, Xero, Sage Intacct, Dynamics 365 Finance

Earnings Transcripts — Accessible through FactSet, Refinitiv, Bloomberg, Capital IQ, or AlphaSense

Google Workspace

ShortcutXL integrates with Google Workspace through the `gws` CLI, giving the agent direct access to your Google account.

- **Gmail** — search, read, and send emails. Pull data from email threads directly into your spreadsheet.
- **Google Drive** — list, search, download, and create files. Access documents, sheets, and other files across your Drive.
- **Google Sheets** — read and write cell ranges in any Google Sheet. Use this to sync data between Google Sheets and Excel.

Outlook & Microsoft 365

ShortcutXL has deep integration with the Microsoft ecosystem through COM automation and local filesystem sync.

- **Outlook** — read and send emails, search your inbox, pull attachments and email data into Excel.
- **OneDrive / SharePoint** — full read and write access to your synced files. No API setup needed — the agent works directly with your local OneDrive and SharePoint sync folders.
- **Microsoft 365** — access to Word, PowerPoint, and other Office documents through COM automation on Windows.

Databases & Cloud Storage

Connect to databases via `/connect`. Credentials are stored in your OS keychain and injected per-query with your explicit approval.

- **Snowflake**, **PostgreSQL**, **Oracle**, **MySQL**

ShortcutXL also has full access to locally synced cloud drives — no API setup needed:

- **Google Drive**, **Dropbox**, **Box** — synced via their respective desktop clients

SaaS & Business Platforms

Connect to APIs via `/connect` with tokens stored securely in your OS keychain.

- **Salesforce** — accounts, contacts, opportunities, leads

- **HubSpot** — contacts, deals, companies
- **Airtable** — tables, records, metadata
- **Notion** — pages, databases, blocks
- **Slack** — messages, channels, users
- **Workday** — HR, finance, planning data
- **SAP** — ERP data

MCP Servers

For tools not covered by built-in skills, ShortcutXL supports MCP (Model Context Protocol) servers. MCP is an open standard that lets the agent connect to any external tool server — GitHub, Gmail, Jira, custom internal tools, and hundreds more.

Ask the agent to set one up ("connect me to GitHub via MCP") or see the MCP docs for manual configuration.

MCP Servers

MCP (Model Context Protocol) lets ShortcutXL connect to external tool servers – GitHub, Slack, Gmail, Postgres, and hundreds more. Any server that speaks the MCP protocol works out of the box.

How It Works

MCP servers are small processes that expose tools over a standard protocol. ShortcutXL launches them at startup, discovers their tools, and makes them available to the agent. You configure which servers to run, the agent handles the rest.

Getting Started

For hosted MCP URLs, use `/mcp` and choose `Add hosted MCP`. ShortcutXL writes the config and starts browser login automatically when the server requires OAuth.

Use streamable HTTP MCP URLs, usually ending in `/mcp`. Do not use legacy `/sse` URLs with the hosted installer.

You can also install from PowerShell:

```
shortcut mcp add notion --url https://mcp.notion.com/mcp
```

Server names are unique. If a name already exists, ShortcutXL rejects the install unless you intentionally replace it with `--force`.

Configuration

Servers are defined in:

```
~/.shortcut/agent/mcp.json
```

Config format

The format is the same as Claude Desktop and Cursor – you can copy-paste configs from any MCP server README:

```
{
  "mcpServers": {
    "github": {
      "command": "pnpm",
      "args": ["dlx", "@modelcontextprotocol/server-github"],
      "connection": "github",
      "startupTimeoutMs": 30000,
      "toolCallTimeoutMs": 600000,
      "resetToolCallTimeoutOnProgress": true
    }
  }
}
```

Timeout settings

ShortcutXL supports separate MCP timeouts for startup and tool execution:

- `startupTimeoutMs`: how long to wait for the server to start and respond to initial handshake. Default: `30000`.
- `toolCallTimeoutMs`: inactivity timeout for a single MCP tool call. Default: `600000` (10 minutes).
- `resetToolCallTimeoutOnProgress`: if `true`, MCP progress notifications reset the inactivity timeout. Default: `true`.
- `maxToolCallTotalTimeoutMs`: optional absolute wall-clock cap for a single tool call. Default: unset.

Legacy `timeout` is still accepted as a startup-timeout alias for older configs.

Credentials

MCP servers need environment variables for authentication (e.g. `GITHUB_TOKEN`). Instead of hardcoding secrets in the config file, reference a `/connect` connection by name:

1. Run `/connect` and create a connection (e.g. `"github"`)
2. Add fields whose names match the env vars the server expects (e.g. `GITHUB_TOKEN`)
3. In `mcp.json`, set `"connection": "github"`

At startup, ShortcutXL looks up the connection in the OS keychain and injects the fields as environment variables into the server process.

For non-sensitive values, use `env` directly in the config:

```
{
  "mcpServers": {
    "slack": {
      "command": "pnpm",
      "args": ["dlx", "@modelcontextprotocol/server-slack"],
      "connection": "slack",
      "env": { "SLACK_TEAM_ID": "T01234" }
    }
  }
}
```

Managing Servers

Use `/mcp` to see connected servers, their tools, and connection status. From there you can also open config files or remove servers.

Servers are loaded at startup — after editing config, restart the session for changes to take effect.

Finding MCP Servers

Browse the official registry at github.com/modelcontextprotocol/servers for a list of available servers. Popular ones include GitHub, Slack, Google Drive, Postgres, Filesystem, and many more.

Extensibility

ShortcutXL is built to be extended. It ships with a rich set of built-in skills, a skill creation system for teaching the agent new workflows, and a full extension API for adding custom tools, commands, and UI.

Built-In Skills

Skills are instruction packages that teach the agent specialized workflows. They're loaded on demand — the agent only reads a skill when your request matches it, so they don't waste context.

SKILL	DESCRIPTION
Integrations	Formula syntax for Bloomberg, FactSet, Capital IQ, Refinitiv, Morningstar, PitchBook, and 15+ other financial data providers and accounting systems
External Services	Connect to databases (Snowflake, PostgreSQL, Oracle, MySQL), APIs (Salesforce, HubSpot, Slack, Notion), Google Workspace, and cloud drives
MCP	Install and configure MCP servers for connecting to external tools like GitHub, Slack, Gmail, and more
Document Creation	Generate PDFs (fpdf2, ReportLab, HTML-to-PDF) and PowerPoint presentations (python-pptx)
SEC EDGAR	Research and download SEC filings (10-K, 10-Q, 8-K) for US public companies, convert to PDF for analysis
Excel Screenshot	Export any Excel range as PNG, JPEG, BMP, or TIFF
Google Workspace	Access Gmail, Google Drive, and Google Sheets
Advanced COM API	Full Excel COM object model reference (~38,700 lines of type stubs) for when the agent needs to do something beyond the standard API

Creating Your Own Skills

You can teach the agent workflows specific to your work by creating skills — just a folder with a `SKILL.md` file:

```
~/.shortcut/agent/skills/my-workflow/SKILL.md
```

```
---
name: quarterly-report
description: Use when the user wants to generate the quarterly financial report.
---

# Quarterly Report Workflow

## Steps
1. Open the template from the shared drive
2. Pull latest data from the finance database
3. Update all pivot tables and charts
```

```
4. Generate the summary sheet with KPIs
5. Export as PDF
```

The `description` field determines when the agent activates the skill — make it specific to the kinds of requests that should trigger it.

Skill Creator

ShortcutXL includes a built-in skill for helping you create other skills. Use `/skill:skill-creator` or ask "help me create a skill" to get a guided workflow that walks you through drafting, testing, and iterating on a new skill — including a benchmark system that compares "with skill" vs "without skill" results so you can objectively measure improvement.

Extensions

Extensions are TypeScript modules that add custom tools, commands, keyboard shortcuts, UI components, and lifecycle hooks to ShortcutXL. The agent can also write extensions for itself at runtime — if it needs a new tool for a task, it can create one.

What Extensions Can Do

Custom Tools — Register new tools the agent can call. Examples from the built-in library:

- **Todo list** — persistent task tracker with custom UI rendering
- **Image generation** — generate images via external APIs and embed them
- **Interactive shell** — run interactive commands (vim, htop) with full terminal support
- **SSH delegation** — delegate all tools to a remote machine
- **Subagent orchestration** — spawn specialized subagents with isolated context windows

Commands & UI — Add slash commands, custom status lines, widgets, overlays, footers, and headers:

- **Presets** — named configurations for model, thinking level, and instructions via `/preset`
- **Tool selector** — interactive `/tools` command to enable/disable tools per session
- **Handoff** — transfer context to a new focused session via `/handoff`
- **Desktop notifications** — alert when the agent finishes a long-running task
- **Custom themes** — sync with OS dark/light mode, custom color schemes

Lifecycle & Safety — Hook into events to add guardrails and automation:

- **Permission gates** — confirm before dangerous commands (rm -rf, sudo)
- **Protected paths** — block writes to sensitive files (.env, .git/)
- **Git checkpoints** — auto-stash at each turn for easy rollback
- **Auto-commit on exit** — commit changes when the session ends

System Prompt Modification — Dynamically modify the agent's behavior by appending to its system prompt at runtime.

Installing Extensions

```
# Load from a file
shortcut --extension path/to/my-extension.ts

# Or drop into the extensions directory for auto-discovery
cp my-extension.ts ~/.shortcut/agent/extensions/
```

Packages

Skills and extensions can be bundled and shared via npm or git:

```
shortcut install npm:@myteam/finance-tools
shortcut install git:github.com/myteam/our-skills
```

```
shortcut list       # See installed packages
shortcut update     # Update all packages
shortcut config     # Enable/disable individual components
```

CLI Reference

Basic Usage

```
shortcut                                # Interactive mode
shortcut "Build a DCF for AAPL"         # Interactive with initial prompt
shortcut -p "Format column D as USD"    # Non-interactive: run and exit
echo "Summarize the workbook" | shortcut  # Pipe stdin (auto print mode)
```

Modes

FLAG	DESCRIPTION
(none)	Interactive TUI — the default experience
`-p` , `--print`	Non-interactive: process the prompt and exit
`--mode json`	NDJSON event stream — for programmatic consumption
`--agent-mode action`	Start in Action mode (default — reads and writes)
`--agent-mode plan`	Start in Plan mode (read-only, explores first, proposes a plan)
`--agent-mode ask`	Start in Ask mode (read-only analysis and audit)

Session Management

Every session is saved automatically. You can continue or resume previous sessions.

FLAG	DESCRIPTION
`-c` , `--continue`	Continue the most recent session
`-r` , `--resume`	Browse and select a past session to resume
`--session <id>`	Continue a specific session by UUID or partial prefix
`--session-dir <dir>`	Custom directory for session storage
`--no-session`	Ephemeral mode — don't save the session

```
shortcut -p "Create a revenue model"
# Output includes: [session:a1b2c3d4-...]

shortcut --session a1b2c3d4 -p "Now add a sensitivity table"
shortcut -c -p "Make the headers blue"
shortcut -r                             # Browse all past sessions
```

Permissions

Runtime filesystem and shell permissions are managed interactively with `/permissions`.

COMMAND / FLAG	DESCRIPTION
`/permissions`	Inspect and change runtime permission settings
`--skip-runtime-permissions`	Bypass runtime filesystem and shell approval checks for that session
`--skip-spreadsheet-permissions`	Allow headless spreadsheet auto-approval for that session

Model & Thinking

FLAG	DESCRIPTION
`--model <id>`	Select an available model
`--model <id>:<level>`	Select an available model with a thinking level
`--thinking <level>`	Set thinking level: `off`, `minimal`, `low`, `medium`, `high`, `xhigh`

Use `/model` inside the CLI to see the currently available models.

File Arguments

Prefix files with `@` to include them in your prompt. Use absolute paths.

```
shortcut -p @C:/Users/you/data.csv "Import this into Excel and create a pivot table"
shortcut -p @requirements.md @template.xlsx "Build this spreadsheet"
```

Resources

FLAG	DESCRIPTION
`--skill <path>`	Load a skill file or directory (repeatable)
`--no-skills`	Disable skill discovery and loading
`-e`, `--extension <path>`	Load an extension file (repeatable)
`--no-extensions`	Disable extension auto-discovery
`--prompt-template <path>`	Load a prompt template (repeatable)

```
shortcut -e ~/my-extension.ts -p "Use my custom tool"
shortcut --skill ~/my-skills/ -p "Run my workflow"
```

Package Management

```
shortcut install npm:@myteam/finance-tools   # Install from npm
shortcut install git:github.com/team/skills   # Install from git
shortcut remove npm:@myteam/finance-tools     # Remove a package
shortcut update                               # Update all packages
shortcut list                                 # List installed packages
shortcut config                               # Enable/disable package resources
shortcut uninstall                            # Remove ShortcutXL entirely
```

Other Flags

FLAG	DESCRIPTION
`--export <file> [out]`	Export a session file to HTML
`--list-models [search]`	List available models with optional search
`--skip-runtime-permissions`	Bypass runtime filesystem and shell approval checks
`--skip-spreadsheet-permissions`	Allow headless spreadsheet auto-approval
`--verbose`	Force verbose startup output
`--offline`	Disable startup network operations
`--dev-mode`	Enable dev mode (AGENTS.md, dev resources, verbose logging)
`-h` , `--help`	Show help
`-v` , `--version`	Show version

Claude Code & Codex Integration

ShortcutXL can be controlled from Claude Code or OpenAI Codex as a plugin. This lets you use ShortcutXL's Excel capabilities from within your existing coding agent workflow — Claude Code handles code and files, ShortcutXL handles Excel.

Installation

Claude Code

```
/plugin marketplace add fundamental-research-labs/shortcutxl-plugins
/plugin install shortcutxl@shortcut
```

Codex

```
$skill-installer install https://github.com/fundamental-research-labs/shortcutxl-plugins/tree/master/shortcutxl
```

Then restart Codex.

How It Works

Once installed, Claude Code (or Codex) gains the ability to call `shortcut` from bash. The plugin teaches the coding agent how to use ShortcutXL — when to invoke it, how to manage sessions, and which mode to use.

```
# Claude Code runs these on your behalf
shortcut -p "Create a DCF model for AAPL"
shortcut --agent-mode plan -p "This workbook is a mess, help me fix it"
shortcut --agent-mode ask -p "Audit this financial model for errors"
```

Session Management

Every `shortcut` invocation outputs a `[session:<uuid>]` line. The plugin teaches the coding agent to remember this UUID and pass it on follow-up calls:

```
# First call — starts a new session
shortcut -p "Create a revenue model"
# Output: [session:a1b2c3d4-...]

# Follow-up — continues the same session
shortcut --session a1b2c3d4 -p "Now add a sensitivity table"
```

The coding agent decides whether a follow-up request should continue the existing session or start a new one. If it's ambiguous, it asks you.

Choosing the Right Mode

The plugin guides the coding agent to pick the right mode automatically:

- **Action** (default) — direct tasks like "create a DCF", "format this table", "add a chart"
- **Plan** (`--agent-mode plan`) — complex or ambiguous requests where the agent should explore the workbook first
- **Ask** (`--agent-mode ask`) — read-only questions like "audit this model", "explain these formulas"

Permissions

ShortcutXL now has an explicit runtime permission surface for file and shell access.

- If a task needs broader file access, the coding agent should tell you to use `/permissions`
- It should not casually disable the boundary with `--skip-runtime-permissions`
- `--skip-runtime-permissions` is only appropriate when you explicitly want runtime filesystem and shell checks bypassed for that session

If the coding agent is discussing the isolated bash environment, that is `/sandbox`, which is separate from `/permissions`.

File Arguments

The coding agent can pass files to ShortcutXL using `@` prefixed absolute paths:

```
shortcut -p @C:/Users/you/data.csv "Import this into Excel and create a pivot table"
shortcut -p @requirements.md @template.xlsx "Build this spreadsheet"
```

What This Enables

With the plugin installed, you can ask Claude Code or Codex to do things like:

- "Open the Q3 financials workbook and build a summary dashboard"
- "Read the CSV in my downloads folder, import it to Excel, and create a pivot table"
- "Audit the financial model in the open workbook for formula errors"
- "Create a PowerPoint from the data in Sheet1, then email it to the team"

The coding agent orchestrates between its own tools (file system, code, git) and ShortcutXL (Excel, Office, data providers) seamlessly.

Security & Privacy

Everything Stays Local

Your workbooks, files, and data stay on your machine. The AI communicates with Excel through a local-only HTTP connection (localhost). No spreadsheet data is sent to external servers beyond what the AI model needs to process your request.

Token Authentication

Each time Excel loads the ShortcutXL add-in, a random 32-byte authentication token is generated and stored locally. Every request from the Shortcut agent to the Excel add-in must include this token. This prevents other programs on your machine from controlling Excel through the add-in.

Change Approval

Every modification the AI makes to your spreadsheet is shown to you before it takes effect:

ADDRESS	BEFORE	AFTER	DETAIL
B2	100	=SUM(B3:B10)	formula
C5		$1,500.00	hardcoded

You can:

- **Accept** — apply the changes
- **Accept All** — auto-approve for the rest of the session
- **Reject** — revert everything to its original state

Read-only operations are auto-approved and never prompt you.

Runtime Permissioning

ShortcutXL also enforces a runtime permission boundary for its built-in file and shell tools.

- Use `/permissions` to review the current boundary
- Add approved workspaces for broad trusted access
- Add narrower file or folder grants outside those workspaces
- Keep spreadsheet approval defaults separate from runtime file and shell permissions

See Permissions for the full runtime permissioning model.

Code Sandbox

Shell commands can optionally run in an isolated environment for stronger security:

Filesystem Isolation

Commands run inside a lightweight Linux container (Alpine Linux on WSL2). By default, the container can only read your home directory and write to the current working directory. All other filesystem access is blocked.

Network Isolation

Three modes, configured in `/settings`:

MODE	BEHAVIOR

None (default)	No network access at all
Proxy	Per-domain approval — you approve each new domain
Full	Unrestricted network access

Write-Path Approval

If the AI needs to write outside the current working directory, you're prompted to approve. You can allow it once or permanently.

Credential Security

When you configure connections to external services (databases, APIs), credentials are stored in your operating system's native keychain:

- **macOS** — Keychain
- **Windows** — Credential Manager

The AI agent cannot read your credentials directly. They are injected as environment variables only into the specific subprocess that needs them, and only with your explicit approval each time. Connection-backed commands always require per-call approval — there is no "Accept All" for credential access.

Execution Sandboxing

Python code that runs inside Excel is sandboxed:

- **Blocked operations** — `import`, `open`, `eval`, `exec`, `compile`, `breakpoint`
- **Network blocked** — no socket access from within Excel execution
- **Subprocess blocked** — no `os.system`, `subprocess.Popen`, or `os.exec`
- **Filesystem writes blocked** — no `open()` with write modes
- **Isolated namespace** — no variable persistence between executions
- **30-second timeout** — long-running code is automatically terminated