DocCompiler.ai

ADMINISTRATOR & DEVELOPER GUIDE

Powertoys Administrator & Developer Guide

Architecture, Configuration, Security & Developer Reference

A Cyber Panda Solutions LLC product

Version 3.1 2026-05-07 @ fde1599 Generated by DocCompiler.ai

Always-updated version at doccompiler.ai/microsoft/PowerToys

Source code analysis by Claude Opus 4.6 | 1M token context

About This Document

What Is This?

This is a comprehensive, source-code-verified **Administrator & Developer Guide** for **Powertoys**, generated by DocCompiler.ai using Claude Opus 4.6 with 1M-token context. Every command, configuration option, and behavior documented here was verified against the actual source code.

The Definitive Reference

This document is designed to be the single source of truth for Powertoys. Rather than piecing together scattered README fragments, outdated wiki pages, forum posts, and version-specific screenshots, you can rely on this as a verified, structured reference backed by automated code review.

Optimized for AI Assistants

This document is structured for consumption by AI assistants. Feed this PDF into your preferred LLM for accurate, grounded answers about Powertoys — instead of having the model guess or hallucinate from incomplete context.

Version & Updates

Generated on **2026-05-07** (version **3.1**). This document reflects the source code at commit `fde1599`. The always-updated version is permanently available at **doccompiler.ai/microsoft/PowerToys**. Do not print or download for offline use — this document is updated when the source code changes.

Our Mission

DocCompiler.ai aims to be the definitive wiki for the world's most important open-source projects. Every document is overkill by design: exhaustive, fact-checked, and built to replace the scattered tribal knowledge that slows teams down.

Table of Contents

Microsoft PowerToys -- Administrator & Developer Guide

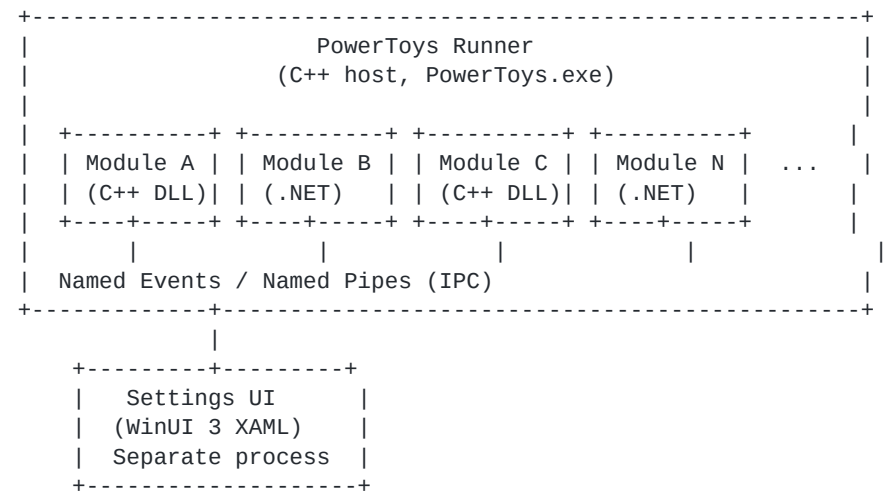
This guide covers the architecture, configuration, deployment, security, and developer internals of Microsoft PowerToys -- a suite of 30+ system utilities for Windows power users. It is written for system administrators managing enterprise deployments, developers contributing to the codebase, and architects evaluating the platform's design.

For everyday usage instructions, keyboard shortcuts, and feature walkthroughs, see the **User Guide**.

1. Architecture Overview

1.1 High-Level Design

PowerToys follows a **host-plugin architecture**. A single native C++ host process (`PowerToys.exe`, the "Runner") manages the lifecycle of independent utility modules. Each module is a self-contained unit -- either a native C++ DLL or a .NET assembly -- loaded dynamically at startup.



The architecture enforces **fault isolation**: a crash in one module does not bring down the Runner or other modules. The Settings UI is a separate WinUI 3 process that communicates with the Runner and individual modules via the filesystem (JSON settings files with file-system watchers).

1.2 Source Layout

```
src/
|-- runner/                 # Runner host process (C++)
|-- settings-ui/            # WinUI 3 settings application
|   |-- Settings.UI/        # Main app project
|   |-- Settings.UI.Library/ # Core models, ViewModels (156 files, 443K chars)
|   +-- QuickAccess.UI/     # Quick settings flyout
```

```
|-- modules/                # All utility modules
|   |-- AdvancedPaste/      # C# (WinUI 3)
|   |-- alwaysontop/        # C++
|   |-- awake/              # C#
|   |-- cmdpal/             # C# (WinUI 3) -- Command Palette
|   |-- colorPicker/        # C# (WinUI)
|   |-- CropAndLock/        # C++
|   |-- EnvironmentVariables/# C# (WinUI)
|   |-- fancyzones/         # C++ + C# CLI
|   |-- Hosts/              # C# (WinUI)
|   |-- imageresizer/       # C# + CLI
|   |-- keyboardmanager/    # C++ (16 files, 96K chars common)
|   |-- launcher/           # C# (PowerToys Run, Wox-based)
|   |-- MouseUtils/         # C++ + C# (Find My Mouse, Highlighter, etc.)
|   |-- MouseWithoutBorders/ # C# (29 core files, 290K chars)
|   |-- NewPlus/            # C++ (shell extension)
|   |-- peek/               # C# (WinUI 3)
|   |-- poweraccent/        # C#
|   |-- powerdisplay/       # C#
|   |-- powerrename/        # C++ (39 files, 250K chars)
|   |-- PowerOCR/           # C# (Text Extractor)
|   |-- previewpane/        # C# (File Explorer add-ons)
|   |-- registrypreview/    # C# (WinUI)
|   |-- ShortcutGuide/      # C++
|   |-- Workspaces/         # C#
|   +-- ZoomIt/             # C++
|-- common/                 # Shared libraries
|   |-- interop/            # C++/C# interop, IPC constants
|   +-- utils/              # 34 files, 176K chars
|-- dsc/                    # DSC v3 provider
|   |-- v3/PowerToys.DSC/   # DSC resource implementation
|   +-- PowerToys.Settings.DSC.Schema.Generator/
|-- update/                 # Auto-update logic
+-- runner/                 # Host entry point
installer/
|-- PowerToysSetup.sln      # WiX installer solution
|-- PowerToysSetupCustomActionsVNext/ # Custom actions (5 files, 64K chars)
+-- generateAllFileComponents.ps1
tools/
|-- BugReportTool/          # 17 files, 58K chars
|-- CleanUp/                # Installation cleanup
|-- MonitorReportTool/      # 7 files, 19K chars
|-- StylesReportTool/       # 5 files, 16K chars
+-- Verification scripts/   # 2 files, 37K chars
```

1.3 Runner Process (`src/runner/`)

The Runner is the central orchestrator:

1. **Startup** -- Parses command-line arguments, initializes COM, creates a hidden message window for system tray notification.
2. **Module Loading** -- Enumerates module manifests, loads each DLL/assembly via the module interface, calls `enable()` on each enabled module.
3. **Auto-Start** -- Creates a Task Scheduler task under `\PowerToys\` so the Runner launches at user logon.
4. **Auto-Update** -- Checks GitHub releases every 24 hours (`UPDATE_CHECK_INTERVAL_MINUTES = 60 * 24`). Downloads updates if the connection is not metered, then stages them via `PowerToys.Update.exe`.

5. **GPO Enforcement** -- Reads Group Policy settings via the `powertoys_gpo` namespace and overrides local user preferences when enterprise policies are set.

> **Important**
>
> The auto-update check interval is hardcoded at 1440 minutes (24 hours). There is no user-facing setting to change this. Enterprise administrators who need to control update rollouts should use GPO to disable auto-update and deploy updates via SCCM/Intune instead.

Update staging process:

```
Stage 1: Runner detects new version on GitHub Releases
         -&gt; Downloads installer to staging directory
         -&gt; Copies PowerToys.Update.exe (action runner) to %TEMP%

Stage 2: PowerToys.Update.exe terminates Runner
         -&gt; Executes the staged installer (MSI)
         -&gt; Runner restarts with new version
```

1.4 Inter-Process Communication (IPC)

Modules communicate with the Runner and with each other through **named Windows events**. Each module has one or more dedicated event names defined as constants in `src/common/interop/Constants.cpp`. When a hotkey is pressed, the Runner signals the appropriate event; the target module's background thread wakes and performs its action.

This design avoids heavyweight IPC (no sockets, no RPC, no shared memory for data). The events are purely signaling mechanisms -- configuration data flows through the filesystem (JSON files with watchers).

Named pipe communication is used between the Runner and the Settings UI for real-time status updates and module enable/disable commands.

1.5 Settings Data Flow

```
User changes setting in Settings UI
  -&gt; Settings UI writes JSON to %LOCALAPPDATA%\Microsoft\PowerToys\{Module}\settings.json
  -&gt; File system watcher in the module detects change (300ms debounce)
  -&gt; Module reloads configuration from disk
  -&gt; Module applies new settings immediately (no restart needed for most settings)
```

General (cross-module) settings live at `%LOCALAPPDATA%\Microsoft\PowerToys\settings.json`. Each module's settings are isolated in a subdirectory named after the module.

2. System Requirements

2.1 Runtime Requirements

Requirement	Minimum	Recommended
OS	Windows 10 v2004 (19041)	Windows 11 23H2+

Architecture	x64, ARM64	x64
.NET	.NET 8 Desktop Runtime	Bundled with installer
Runtime	VC++ Redistributable 2015-2022	Bundled with installer
WebView2	Evergreen Runtime	Bundled with installer
Disk	~400 MB	~500 MB (with all modules)
RAM	100 MB baseline	200-500 MB depending on active modules
Privileges	Standard user (most modules)	Admin for Hosts File Editor, Environment Variables (system-level)

2.2 Build Requirements

Component	Version
Visual Studio	2022 17.8+ or 2026
C++ Workload	Desktop Development with C++
.NET Workload	.NET 8 SDK + .NET Desktop Development
Windows SDK	10.0.26100.0 or later
Node.js	Current LTS (for Monaco editor components)
WiX Toolset	4.x (for installer builds)
Platform targets	x64, ARM64

2.3 Module-Specific Requirements

Module	Additional Requirements
Advanced Paste	OpenAI API key (for AI-powered paste)
Command Not Found	PowerShell 7.4+, WinGet
Text Extractor (PowerOCR)	Windows OCR language packs
Mouse Without Borders	Network connectivity between machines
File Explorer Add-ons	Windows Explorer process restart after install
ZoomIt	Screen recording requires Windows Graphics Capture API

> **Note**
> The installer bundles .NET 8 Desktop Runtime, VC++ Redistributable, and WebView2 Runtime. Enterprise deployments that already have these runtimes deployed can use the per-user installer to reduce package size.

3. Configuration Reference

3.1 Settings File Locations

All settings are stored as JSON files under the user's local application data directory:

```
%LOCALAPPDATA%\Microsoft\PowerToys\
|-- settings.json                    # General settings (startup, theme, etc.)
|-- ColorPicker\settings.json        # Color Picker module settings
|-- FancyZones\
|   |-- settings.json                # FancyZones module settings
|   |-- applied-layouts.json         # Currently applied zone layouts
|   |-- custom-layouts.json          # User-defined layouts
|   |-- layout-hotkeys.json          # Layout switching hotkeys
|   +-- layout-templates.json        # Built-in layout templates
|-- Keyboard Manager\
|   |-- default.json                 # Key remapping configuration
|   +-- settings.json
|-- PowerToys Run\settings.json
|-- ImageResizer\settings.json
|-- PowerRename\settings.json
+-- ... (one directory per module)
```

3.2 General Settings (`settings.json`)

json

```
{
  "general": {
    "startup": true,
    "enabled": {
      "ColorPicker": true,
      "FancyZones": true,
      "File Locksmith": true,
      "Hosts File Editor": true,
      "Image Resizer": true,
      "Keyboard Manager": true,
      "Find My Mouse": true,
      "Mouse Highlighter": true,
      "Mouse Jump": true,
      "Mouse Pointer Crosshairs": true,
      "Power Rename": true,
      "PowerToys Run": true,
      "Shortcut Guide": true,
      "Advanced Paste": true,
      "Peek": true,
      "Command Palette": true,
      "Workspaces": true,
      "ZoomIt": true
    },
```

```
    "theme": "system",
    "run_elevated": false,
    "download_updates_automatically": true,
    "show_new_updates_toast_notification": true
  }
}
```

3.3 Module Configuration Deep Dives

Color Picker

json

```
{
  "properties": {
    "ActivationShortcut": {
      "win": false, "ctrl": true, "alt": false, "shift": false,
      "code": 19, "key": "Break"
    },
    "ColorHistoryLimit": 20,
    "VisibleColorFormats": {
      "HEX": true, "RGB": true, "HSL": true, "HSV": false,
      "CMYK": false, "HSB": false, "HSI": false, "HWB": false,
      "NCol": false, "CIELAB": false, "CIEXYZ": false,
      "VEC4": false, "Decimal": false
    },
    "ActivationAction": "OpenColorPickerAndThenEditor",
    "CopiedColorRepresentation": "HEX",
    "ShowColorName": true
  }
}
```

Default activation: **Ctrl+Break**. The `ActivationAction` supports three modes: `OpenEditor` (editor only), `OpenColorPickerAndThenEditor` (pick then edit), and `ColorPickerOnly` (pick and copy directly).

Awake

json

```
{
  "properties": {
    "keepDisplayOn": true,
    "mode": "INDEFINITE",
    "hours": 0,
    "minutes": 30,
    "expirationDateTime": "2026-05-07T18:00:00Z"
  }
}
```

Mode	Behavior
`PASSIVE`	Awake disabled, OS manages sleep normally
`INDEFINITE`	System stays awake until manually disabled
`TIMED`	Stays awake for `hours`:`minutes` then reverts to PASSIVE

`EXPIRABLE`	Stays awake until `expirationDateTime` then reverts

> **Warning**
>
> When Awake has `keepDisplayOn: true`, it calls `SetThreadExecutionState` with `ES_DISPLAY_REQUIRED`, which blocks Task Scheduler idle detection entirely (GitHub issue #44134). Scheduled tasks configured to run "only when idle" will never trigger while Awake is active with display-on. Enterprise environments with idle-triggered maintenance tasks should either disable the display-on flag or use GPO to prevent users from enabling it.

FancyZones

FancyZones uses multiple configuration files for layout management:

`applied-layouts.json` -- maps monitor configurations to zone layouts:

```json
{
  "applied-layouts": [
    {
      "device": {
        "monitor": "DELA103",
        "monitor-instance": "DISPLAY\\DELA103\\{guid}",
        "serial-number": "ABC123",
        "monitor-number": 1,
        "virtual-desktop": "{desktop-guid}"
      },
      "applied-layout": {
        "uuid": "{layout-guid}",
        "type": "custom"
      }
    }
  ]
}
```

`custom-layouts.json` -- user-defined zone arrangements:

```json
{
  "custom-layouts": [
    {
      "uuid": "{layout-guid}",
      "name": "Development Layout",
      "type": "canvas",
      "info": {
        "ref-width": 3840,
        "ref-height": 2160,
        "zones": [
          { "X": 0, "Y": 0, "width": 1920, "height": 2160 },
          { "X": 1920, "Y": 0, "width": 1920, "height": 1080 },
          { "X": 1920, "Y": 1080, "width": 1920, "height": 1080 }
        ],
        "sensitivity-radius": 20
      }
    }
  ]
```

```
}
```

Known issue: The "Move Newly Created Windows" feature in FancyZones can snap child windows (dialogs, popups) into zones, pulling them away from their parent window (GitHub issues #35688, #38084). If applications rely on modal dialogs, consider disabling this feature.

Keyboard Manager

Remappings are stored in `%LOCALAPPDATA%\Microsoft\PowerToys\Keyboard Manager\default.json`:

```json
{
  "remapKeys": {
    "inProcess": [
      { "originalKeys": "162", "newRemapKeys": "91" }
    ]
  },
  "remapShortcuts": {
    "global": [
      {
        "originalKeys": "164;65",
        "newRemapKeys": "162;65",
        "runProgram": "",
        "openUri": "",
        "operationType": 0,
        "secondKeyOfChord": 0,
        "targetApp": ""
      }
    ],
    "appSpecific": [
      {
        "originalKeys": "162;87",
        "newRemapKeys": "162;160;87",
        "targetApp": "code"
      }
    ]
  }
}
```

Key codes use Windows virtual key codes. The `targetApp` field enables per-application shortcut remapping, matching by process name (without `.exe`).

3.4 Configuration-as-Code: DSC v3

PowerToys includes a Windows DSC (Desired State Configuration) v3 provider for managing settings declaratively. This is the recommended approach for enterprise deployments.

Provider location: `src/dsc/v3/PowerToys.DSC/`

Supported commands:

Command	Purpose
`get`	Retrieve current settings state
`set`	Apply desired settings state

`test`	Check if current state matches desired
`export`	Export current configuration
`schema`	Output JSON schema for the resource
`manifest`	Output DSC resource manifest
`modules`	List available PowerToys modules

Command-line options:

Option	Required	Description
`--resource`	Yes	Resource type (currently only `settings`)
`--module`	No	Target PowerToys module name
`--input`	No	JSON state to apply

Example: Configure FancyZones via DSC

yaml

```yaml
# powertoys-config.dsc.yaml
properties:
  resources:
    - resource: Microsoft.PowerToys.Configure/PowerToysConfigure
      directives:
        description: Configure FancyZones
      settings:
        FancyZones:
          Enabled: true
          FancyzonesShiftDrag: true
          FancyzonesMouseDrag: false
          FancyzonesOverrideSnapHotkeys: true
          FancyzonesZoneHighlightColor: "#0078D4"
          FancyzonesEditorHotkey:
            win: true
            ctrl: false
            alt: false
            shift: false
            code: 192
            key: "`"
```

Apply with:

powershell

```powershell
winget configure powertoys-config.dsc.yaml
```

> **Tip**
>
> The DSC provider's `export` command is useful for capturing a "known-good" configuration from a reference machine, which can then be applied across a fleet. Run `PowerToys.DSC.exe export --resource settings` to dump the complete current state.

The schema generator at `src/dsc/PowerToys.Settings.DSC.Schema.Generator/` produces a JSON schema that documents every configurable property across all modules. This schema powers IntelliSense in editors that support DSC authoring.

4. Deployment & Scaling

4.1 Installation Methods

PowerToys is a **desktop application** -- there is no server-side deployment, container image, or cloud component.

Method	Scope	Best For
GitHub Releases (`.exe`)	Per-user or per-machine	Individual developers
Microsoft Store	Per-user	Automatic updates via Store
WinGet	Per-user or per-machine	CLI-driven installs
MSI (WiX-built)	Per-machine	Enterprise SCCM/Intune
DSC v3	Per-user settings	Fleet configuration management

WinGet installation:

```powershell
# Install latest
winget install Microsoft.PowerToys --source winget

# Install specific version
winget install Microsoft.PowerToys --version 0.87.0 --source winget

# Silent install (per-machine)
winget install Microsoft.PowerToys --scope machine --silent
```

4.2 Enterprise Deployment via Intune/SCCM

For large-scale deployment:

1. **Download the MSI** from GitHub Releases (not the `.exe` bootstrapper).
2. **Pre-install runtimes** if not already deployed: .NET 8 Desktop Runtime, VC++ 2015-2022 Redistributable, WebView2 Evergreen Runtime.
3. **Deploy the MSI** via your management tool with silent flags:

```powershell
msiexec /i PowerToysSetup-&lt;version&gt;-x64.msi /quiet /norestart
```

4. **Apply configuration** via DSC or by distributing pre-configured `settings.json` files to `%LOCALAPPDATA%\Microsoft\PowerToys\`.
5. **Lock settings** via Group Policy (see Section 7.2).

4.3 Installer Internals

The installer is built with WiX Toolset. Key details:

- **Component organization**: Each module has a separate `.wxs` file in `installer/`, enabling granular control over which modules are installed.
- **File enumeration**: The PowerShell script `generateAllFileComponents.ps1` scans build output and generates WiX component definitions automatically.
- **Custom actions** (`installer/PowerToysSetupCustomActionsVNext/`, 5 files, 64K chars): Handle Task Scheduler registration, shell extension registration, and cleanup of previous installations.
- **Platform targets**: x64 and ARM64 are built as separate MSI packages.
- **Shell extensions registered**: Image Resizer, Power Rename, File Locksmith, and New+ register Explorer context menu entries.
- **Registry**: Module registration under `HKCU\Software\Classes\powertoys\components`.

4.4 Auto-Start Mechanism

The Runner creates a Task Scheduler task under `\PowerToys\` at installation:

xml

```
&lt;!-- Simplified representation of the scheduled task --&gt;
&lt;Task&gt;
  &lt;Triggers&gt;
    &lt;LogonTrigger&gt;
      &lt;UserId&gt;{current-user-sid}&lt;/UserId&gt;
    &lt;/LogonTrigger&gt;
  &lt;/Triggers&gt;
  &lt;Actions&gt;
    &lt;Exec&gt;
      &lt;Command&gt;"%LOCALAPPDATA%\PowerToys\PowerToys.exe"&lt;/Command&gt;
    &lt;/Exec&gt;
  &lt;/Actions&gt;
  &lt;Settings&gt;
    &lt;RunLevel&gt;LeastPrivilege&lt;/RunLevel&gt;
  &lt;/Settings&gt;
&lt;/Task&gt;
```

The "Run at startup" toggle in Settings UI creates or removes this task. When `run_elevated` is true, the task is configured with `HighestAvailable` run level.

> **Caution**
>
> Running PowerToys elevated (`run_elevated: true`) grants all modules full administrator privileges. This is rarely necessary -- only Hosts File Editor and Environment Variables (system-level) benefit from elevation. The recommended approach is to keep PowerToys running at standard privilege and let individual modules prompt for elevation when needed via UAC.

5. Operations

5.1 Logging and Diagnostics

PowerToys uses **ETW (Event Tracing for Windows)** for telemetry and diagnostics. The provider is `Microsoft.PowerToys.Telemetry`.

Capturing a trace:

```powershell
# Start trace capture
wpr.exe -start "PowerToys.wprp"

# Reproduce the issue, then stop
wpr.exe -stop "Trace.etl"

# Analyze with Windows Performance Analyzer (WPA) or tracerpt
tracerpt Trace.etl -o trace_report.xml
```

Each module emits structured events for:

- Module activation/deactivation
- Settings changes
- Command execution (e.g., FancyZones CLI emits `FancyZonesCLICommandEvent` with command name and success status)
- Errors and exceptions

Module-level logs are written to `%LOCALAPPDATA%\Microsoft\PowerToys\{ModuleName}\Logs\`.

5.2 Bug Report Tool

The built-in Bug Report Tool (`tools/BugReportTool/`, 17 files, 58K chars) collects:

- System information (OS version, architecture, display configuration)
- PowerToys version and enabled modules
- Recent logs from all modules
- Configuration files (sanitized)
- ETW trace snippets

Generate a bug report from Settings UI > General > Bug Report, or run the tool directly:

```powershell
&amp;amp; "$env:LOCALAPPDATA\PowerToys\tools\BugReportTool.exe"
```

5.3 Diagnostic Tools

Tool	Location	Purpose
BugReportTool	`tools/BugReportTool/`	Comprehensive diagnostic bundle
MonitorReportTool	`tools/MonitorReportTool/` (7 files, 19K chars)	Display/monitor enumeration and DPI reporting
StylesReportTool	`tools/StylesReportTool/` (5 files, 16K chars)	Window style flags for debugging overlay/topmost issues
CleanUp	`tools/CleanUp/`	Remove leftover files from failed installs

Module Loader	`tools/` (9 files, 69K chars)	Test module loading in isolation

5.4 Common Operational Issues

Module fails to load:

Check that the module DLL exists in the installation directory and that required runtimes are installed. The `.dll` registration error for Image Resizer (#34745) is typically caused by a missing VC++ Redistributable or corrupted installation -- repair or reinstall resolves it.

Settings not persisting:

Verify write permissions to `%LOCALAPPDATA%\Microsoft\PowerToys\`. Folder redirection or roaming profile configurations can interfere with the file watchers. The 300ms debounce on file watchers means rapid successive changes may be coalesced.

Awake not honoring settings:

If Awake appears to ignore mode changes (#35572, 15 comments), check for conflicting power management software. Some enterprise power management agents override `SetThreadExecutionState` calls. Verify in Event Viewer under `System > Power-Troubleshooter`.

Mouse Utilities cannot be disabled:

This issue (#34524) can occur when the Runner loses its handle to the module's named event. Restart PowerToys completely (exit from system tray, then relaunch).

Command Not Found module issues:

Several known issues affect this PowerShell module:

- Requires WinGet to be installed and functional (#31350)
- May show "untrusted repository" warning for the WinGet module source (#31378)
- Can cause PowerShell startup failures without internet connectivity (#33669)
- May throw PowerShell exceptions during lookup (#33304, 10 comments)

Keyboard Manager errors:

Complex remapping configurations can trigger errors (#34798, 37 comments). Common causes include circular remappings, conflicts with system-reserved keys (Ctrl+Alt+Del), and remappings targeting processes that run elevated when PowerToys is not.

5.5 Health Monitoring

There is no built-in health endpoint (PowerToys is a desktop application, not a service). For enterprise monitoring:

1. **Process monitoring**: Check for `PowerToys.exe` running in user sessions.
2. **Version auditing**: Read the registry at `HKCU\Software\Classes\powertoys\` or parse the installed MSI version.
3. **Configuration compliance**: Use DSC `test` command to verify settings match desired state.

```
powershell
```

```
# Check if PowerToys is running
Get-Process -Name "PowerToys" -ErrorAction SilentlyContinue

# Check installed version via registry
Get-ItemProperty "HKCU:\Software\Classes\powertoys" -ErrorAction SilentlyContinue

# Or for MSI installs
Get-Package -Name "*PowerToys*" -ProviderName msi -ErrorAction SilentlyContinue
```

6. Performance

6.1 Resource Footprint

The Runner process itself is lightweight (~15-30 MB RAM). Each module adds to the footprint based on its complexity:

Module Category	Typical RAM Impact	CPU Impact
Passive hooks (Always on Top, Shortcut Guide)	5-15 MB each	Negligible until activated
Background services (Awake, FancyZones)	10-30 MB each	Low (event-driven)
UI-heavy modules (Command Palette, Peek, Settings)	50-150 MB when open	Moderate during rendering
Mouse Without Borders	30-80 MB	Low-moderate (network I/O)
File Explorer Add-ons	Per Explorer process	Proportional to preview file size

6.2 Startup Impact

PowerToys uses Task Scheduler for auto-start, which runs after the user's desktop is loaded. Module loading is sequential during Runner initialization. On a typical system:

- **Cold start** (first boot): 3-8 seconds for all modules to initialize
- **Warm start** (Runner restart): 1-3 seconds

Disabling unused modules reduces startup time and memory footprint proportionally.

6.3 File Explorer Add-on Performance

Preview handlers run inside the Explorer process (`explorer.exe`). Poorly performing preview handlers can cause Explorer to hang. The PowerToys preview handlers support:

Format	Handler	Notes
SVG	SVG Preview	WebView2-based rendering
Markdown	Markdown Preview	WebView2-based with syntax highlighting

PDF	PDF Preview	Windows built-in or custom renderer
G-code	G-code Preview	3D model visualization
BGcode	BGcode Preview	Binary G-code variant
QOI	QOI Preview	Quite OK Image format
Developer files	DevFiles Preview	Source code with syntax highlighting

> **Tip**
> If File Explorer becomes sluggish after enabling preview handlers, check whether large files (>50 MB) are triggering preview rendering. Disabling previews for specific file types can be done by toggling individual handlers in Settings UI > File Explorer Add-ons.

6.4 FancyZones Performance

FancyZones hooks into window management messages (WM_MOVING, WM_SIZING). On multi-monitor setups with many zones:

- Zone calculation is O(n) per monitor per drag event
- Custom canvas layouts with >20 zones per monitor may introduce perceptible lag during drag operations
- The `sensitivity-radius` setting in custom layouts controls the snap detection area -- larger values increase CPU usage during drags

6.5 Mouse Hook Performance

The low-level mouse and keyboard hooks installed by Mouse Utilities and Keyboard Manager add a small latency to every input event system-wide. In normal operation this is sub-millisecond, but:

- Running antivirus real-time scanning alongside input hooks can compound latency
- Mouse Pointer Crosshairs (#24096) can exhibit position drift when using precision touchpads due to the coordinate translation overhead
- Disabling unused Mouse Utilities sub-modules (Find My Mouse, Highlighter, Crosshairs, Jump) reduces hook overhead

7. Security

7.1 Privilege Model

PowerToys follows the principle of least privilege:

Component	Default Privilege	Elevation Needed For
Runner	Standard user	N/A (unless `run_elevated`)
Settings UI	Standard user	N/A

Hosts File Editor	Standard user	Writing to `C:\Windows\System32\drivers\etc\hosts`
Environment Variables	Standard user	Modifying system-level variables
Keyboard Manager	Standard user	Remapping in elevated windows
All other modules	Standard user	N/A

Modules that need elevation use separate events. For example, Hosts File Editor defines both `SHOW_HOSTS_EVENT` (standard) and `SHOW_HOSTS_ADMIN_EVENT` (elevated), and Environment Variables defines `SHOW_ENVIRONMENT_VARIABLES_EVENT` and `SHOW_ENVIRONMENT_VARIABLES_ADMIN_EVENT`. The Runner spawns an elevated helper process only when the admin event is triggered.

7.2 Group Policy (GPO) Support

Enterprise administrators can control PowerToys behavior via Group Policy. The GPO integration is implemented in `src/common/utils/gpo.h` and the `powertoys_gpo` namespace.

Policy categories:

Policy	Effect
Enable/disable individual modules	Prevents users from toggling specific modules
Disable auto-update	Prevents the Runner from checking GitHub for updates
Disable data diagnostics	Prevents ETW telemetry collection
Force-enable/disable specific features	Per-module feature control
Installer policies	Control per-user vs. per-machine installation

GPO settings override user preferences in `settings.json`. When a policy is enforced, the corresponding toggle in the Settings UI is grayed out with a "Managed by your organization" indicator.

Registry-based GPO paths:

```
HKLM\SOFTWARE\Policies\PowerToys\
|-- AllowAutoUpdate          (DWORD: 0 = disabled, 1 = enabled)
|-- AllowDataDiagnostics     (DWORD: 0 = disabled, 1 = enabled)
|-- Modules\
|   |-- FancyZones           (DWORD: 0 = force-off, 1 = force-on, not set = user choice)
|   |-- PowerToysRun         (DWORD: ...)
|   |-- CommandPalette       (DWORD: ...)
|   +-- ... (one per module)
```

> **Important**
>
> GPO policies are read at Runner startup and cached. Changes to GPO settings require a PowerToys restart to take effect. The Settings UI polls for GPO changes periodically, but the module enforcement happens only during Runner initialization.

7.3 Input Handling Security

Several modules intercept keyboard and mouse input system-wide:

- **Keyboard Manager**: Installs a low-level keyboard hook (`WH_KEYBOARD_LL`) to intercept and remap keystrokes. This hook runs in the Runner's process context.
- **Mouse Utilities**: Installs mouse hooks (`WH_MOUSE_LL`) for Find My Mouse, Highlighter, Crosshairs, and Cursor Wrap.
- **PowerToys Run / Command Palette**: Registers global hotkeys via `RegisterHotKey()`.

These hooks can trigger false positives from endpoint security products. Whitelist the following processes in your EDR/antivirus:

```
%LOCALAPPDATA%\PowerToys\PowerToys.exe
%LOCALAPPDATA%\PowerToys\modules\*
```

7.4 Shell Extension Security

Four modules register shell extensions (Explorer context menu handlers):

Module	Extension Type	Registration
Image Resizer	Shell context menu	`PowerToys.ImageResizerExt.dll`
Power Rename	Shell context menu	`PowerRenameExt.dll`
File Locksmith	Shell context menu	`PowerToys.FileLocksmithExt.dll`
New+	Shell context menu	`PowerToys.NewPlus.dll`

These DLLs run inside the `explorer.exe` process space. A vulnerability in any shell extension could potentially be exploited within the Explorer context. Microsoft signs all release binaries to mitigate tampering.

7.5 Network Exposure

PowerToys is a local desktop application with minimal network surface:

Component	Network Activity	Destination
Auto-update	HTTPS GET	`api.github.com`, `github.com`
Advanced Paste (AI)	HTTPS POST	OpenAI API (`api.openai.com`)
Mouse Without Borders	TCP/UDP	LAN peers (direct connection)
Command Not Found	HTTPS	WinGet package index
WebView2 previews	HTTPS	Content-dependent

Mouse Without Borders is the only module that listens on network ports. It communicates between machines on the local network for keyboard/mouse sharing. Traffic is encrypted but the protocol is proprietary.

> **Warning**
> Advanced Paste sends clipboard content to the OpenAI API when AI-powered formatting is used. Ensure enterprise data loss prevention (DLP) policies account for this. If clipboard data may contain sensitive information, disable Advanced Paste's AI features via GPO or remove the OpenAI API key from the module's configuration.

7.6 Update Security

Updates are downloaded from GitHub Releases over HTTPS. The installer is code-signed by Microsoft. The auto-update mechanism does not perform certificate pinning beyond standard Windows TLS validation. The staged update (`PowerToys.Update.exe`) validates the downloaded MSI signature before execution.

8. Developer Internals

8.1 Module Interface

Every PowerToys module implements a standard interface that the Runner uses for lifecycle management:

C++ modules export these functions from their DLL:

cpp

```cpp
// Module interface (simplified from source)
extern "C" {
    // Called when the module is loaded
    void enable();

    // Called when the module is disabled
    void disable();

    // Called to check if the module is currently enabled
    bool is_enabled();

    // Called when PowerToys is shutting down
    void destroy();

    // Returns the module's display name
    const wchar_t* get_name();

    // Returns the module's unique key
    const wchar_t* get_key();
}
```

.NET modules implement an equivalent managed interface and are loaded via the CLR hosting API. The interop layer in `src/common/interop/` bridges the C++ Runner with .NET module assemblies.

8.2 Named Event System (Complete Reference)

The following tables document every named event constant defined in `src/common/interop/Constants.cpp`. These are the primary IPC mechanism between the Runner and modules.

Launcher and Command Modules:

Event Constant	Module	Purpose
POWER_LAUNCHER_SHARED_EVENT	PowerToys Run	Activate launcher
CMDPAL_SHOW_EVENT	Command Palette	Show Command Palette

Visual Utilities:

Event Constant	Module	Purpose
SHOW_COLOR_PICKER_SHARED_EVENT	Color Picker	Activate color picker
TERMINATE_COLOR_PICKER_SHARED_EVENT	Color Picker	Shut down color picker
ALWAYS_ON_TOP_PIN_EVENT	Always on Top	Toggle pin on focused window
SHORTCUT_GUIDE_TRIGGER_EVENT	Shortcut Guide	Show Windows shortcuts overlay
MEASURE_TOOL_TRIGGER_EVENT	Screen Ruler	Activate pixel measurement
SHOW_PEEK_SHARED_EVENT	Peek	Open quick preview
TERMINATE_PEEK_SHARED_EVENT	Peek	Close Peek
SHOW_POWEROCR_SHARED_EVENT	Text Extractor	Activate OCR capture
TERMINATE_POWEROCR_SHARED_EVENT	Text Extractor	Terminate OCR
REGISTRY_PREVIEW_TRIGGER_EVENT	Registry Preview	Open .reg file viewer

Mouse Utilities:

Event Constant	Module	Purpose
FIND_MY_MOUSE_TRIGGER_EVENT	Find My Mouse	Highlight cursor location
MOUSE_HIGHLIGHTER_TRIGGER_EVENT	Mouse Highlighter	Toggle click highlighting
MOUSE_CROSSHAIRS_TRIGGER_EVENT	Mouse Pointer Crosshairs	Toggle crosshairs
CURSOR_WRAP_TRIGGER_EVENT	Cursor Wrap	Enable screen edge wrapping
MOUSE_JUMP_SHOW_PREVIEW_EVENT	Mouse Jump	Show jump grid
TERMINATE_MOUSE_JUMP_SHARED_EVENT	Mouse Jump	Close jump grid

Window Management:

Event Constant	Module	Purpose
FANCY_ZONES_EDITOR_TOGGLE_EVENT	FancyZones	Open zone editor
FZE_EXIT_EVENT	FancyZones	Close zone editor
CROP_AND_LOCK_THUMBNAIL_EVENT	Crop and Lock	Create thumbnail crop

`CROP_AND_LOCK_REPARENT_EVENT`	Crop and Lock	Reparent window region
`CROP_AND_LOCK_SCREENSHOT_EVENT`	Crop and Lock	Screenshot-based crop
`WORKSPACES_LAUNCH_EDITOR_EVENT`	Workspaces	Open workspace editor
`WORKSPACES_HOTKEY_EVENT`	Workspaces	Trigger workspace restore

System Utilities:

Event Constant	Module	Purpose
`SHOW_HOSTS_EVENT`	Hosts File Editor	Open editor (standard)
`SHOW_HOSTS_ADMIN_EVENT`	Hosts File Editor	Open editor (elevated)
`TERMINATE_HOSTS_EVENT`	Hosts File Editor	Close editor
`SHOW_ENVIRONMENT_VARIABLES_EVENT`	Environment Variables	Open editor (standard)
`SHOW_ENVIRONMENT_VARIABLES_ADMIN_EVENT`	Environment Variables	Open editor (elevated)
`AWAKE_EXIT_EVENT`	Awake	Terminate Awake
`LIGHTSWITCH_TOGGLE_EVENT`	Light Switch	Toggle light/dark mode

Advanced Paste:

Event Constant	Module	Purpose
`ADVANCED_PASTE_SHOW_UI_EVENT`	Advanced Paste	Open paste UI
Various `ADVANCED_PASTE_*_MESSAGE`	Advanced Paste	Format-specific paste triggers

Display and Zoom:

Event Constant	Module	Purpose
`TOGGLE_POWER_DISPLAY_EVENT`	Power Display	Toggle display profile
`TERMINATE_POWER_DISPLAY_EVENT`	Power Display	Shut down Power Display
`REFRESH_POWER_DISPLAY_MONITORS_EVENT`	Power Display	Refresh monitor list
`ZOOMIT_ZOOM_EVENT`	ZoomIt	Activate zoom mode
`ZOOMIT_DRAW_EVENT`	ZoomIt	Activate drawing mode
`ZOOMIT_BREAK_EVENT`	ZoomIt	Start break timer
`ZOOMIT_LIVEZOOM_EVENT`	ZoomIt	Activate live zoom
`ZOOMIT_SNIP_EVENT`	ZoomIt	Activate snip mode
`ZOOMIT_RECORD_EVENT`	ZoomIt	Start screen recording

Other Modules:

Event Constant	Module	Purpose
`POWERACCENT_EXIT_EVENT`	Quick Accent	Terminate Quick Accent
`MWB_TOGGLE_EASY_MOUSE_EVENT`	Mouse Without Borders	Toggle easy mouse mode
`MWB_RECONNECT_EVENT`	Mouse Without Borders	Reconnect to peers

Preview handler resize events are also defined for G-code, BGcode, QOI, DevFiles, Markdown, PDF, and SVG, controlling rendering behavior in the File Explorer preview pane.

8.3 Complete Module Inventory

#	Module	Source Path	Language	Description
1	Advanced Paste	`src/modules/AdvancedPaste/`	C# (WinUI 3)	AI-powered clipboard formatting via OpenAI/Semantic Kernel
2	Always on Top	`src/modules/alwaysontop/`	C++	Pin windows above others via WS_EX_TOPMOST
3	Awake	`src/modules/awake/`	C#	Prevent system sleep via SetThreadExecutionState
4	Color Picker	`src/modules/colorPicker/`	C# (WinUI)	Screen color sampling with format conversion
5	Command Not Found	PowerShell module	PowerShell	WinGet package suggestions for unknown commands
6	Command Palette	`src/modules/cmdpal/`	C# (WinUI 3)	Extensible command launcher with plugin system
7	Crop and Lock	`src/modules/CropAndLock/`	C++	Window region cropping and isolation
8	Cursor Wrap	`src/modules/MouseUtils/CursorWrap/`	C++	Screen edge cursor teleportation
9	Environment Variables	`src/modules/EnvironmentVariables/`	C# (WinUI)	GUI environment variable editor with profiles
10	FancyZones	`src/modules/fancyzones/`	C++ + C# CLI	Window zone layout management
11	File Explorer Add-ons	`src/modules/previewpane/`	C#	Preview and thumbnail handlers for Explorer
12	File Locksmith	`src/modules/FileLocksmith/`	C# (WinUI)	Process file lock identification

13	Hosts File Editor	src/modules/Hosts/	C# (WinUI)	GUI hosts file editor with elevation support
14	Image Resizer	src/modules/imageresizer/	C# + CLI	Batch image resizing with preset dimensions
15	Keyboard Manager	src/modules/keyboardmanager/	C++ (16 files, 96K chars)	Low-level keyboard hook key/shortcut remapping
16	Light Switch	Service module	C#	System appearance mode toggle (light/dark)
17	Find My Mouse	src/modules/MouseUtils/FindMyMouse/	C++	Double-Ctrl cursor spotlight highlight
18	Mouse Highlighter	src/modules/MouseUtils/MouseHighlighter/	C++	Click visualization overlay
19	Mouse Jump	src/modules/MouseUtils/MouseJump/	C#	Large-display cursor teleportation grid
20	Mouse Pointer Crosshairs	src/modules/MouseUtils/MousePointerCrosshairs/	C++	Full-screen crosshair cursor overlay
21	Mouse Without Borders	src/modules/MouseWithoutBorders/	C# (29 core files, 290K chars)	Multi-machine keyboard/mouse sharing
22	New+	src/modules/NewPlus/	C++ (shell extension)	Context menu file creation from templates
23	Peek	src/modules/peek/	C# (WinUI 3)	Quick file preview without opening apps
24	Power Display	src/modules/powerdisplay/	C#	Monitor display profile management
25	Power Rename	src/modules/powerrename/	C++ (39 files, 250K chars)	Regex-powered bulk file rename
26	PowerToys Run	src/modules/launcher/	C#	Wox-based quick launcher with plugins
27	Quick Accent	src/modules/poweraccent/	C#	Accented character insertion overlay
28	Registry Preview	src/modules/registrypreview/	C# (WinUI)	.reg file visual viewer and editor
29	Screen Ruler	via MeasureTool	C++	On-screen pixel measurement tool
30	Shortcut Guide	src/modules/ShortcutGuide/	C++	Windows keyboard shortcuts overlay
31	Text Extractor	src/modules/PowerOCR/	C#	OCR-based on-screen text extraction

32	Workspaces	src/modules/Workspaces/	C#	Window arrangement save and restore
33	ZoomIt	src/modules/ZoomIt/	C++	Screen zoom, annotation, recording, and break timer

8.4 Build System

The solution is organized as a multi-project Visual Studio solution:

```
PowerToys.sln
|-- src/runner/                     # C++ project -&gt; PowerToys.exe
|-- src/settings-ui/Settings.UI/    # C# WinUI 3 -&gt; PowerToys.Settings.exe
|-- src/modules/*/                  # One project per module
|-- src/common/                     # Shared static libs and assemblies
+-- installer/PowerToysSetup/       # WiX installer project
```

Building from source:

powershell

```
# Clone the repository
git clone https://github.com/microsoft/PowerToys.git
cd PowerToys
git submodule update --init --recursive

# Build from Developer Command Prompt for VS 2022+
msbuild PowerToys.sln /p:Configuration=Release /p:Platform=x64

# Or open PowerToys.sln in Visual Studio and build from IDE
```

Build configurations:

Configuration	Purpose
`Debug\	x64`
`Release\	x64`
`Release\	ARM64`

Installer build:

powershell

```
cd installer

# Generate WiX component files from build output
powershell -File generateAllFileComponents.ps1

# Build installer
msbuild PowerToysSetup.sln /p:Configuration=Release /p:Platform=x64
```

8.5 Testing Strategy

PowerToys uses a multi-layered testing approach:

Test Type	Framework	Location	Scope
C++ Unit Tests	CppUnitTest (MSTest)	`src/modules/*/tests/`	Individual C++ module logic
C# Unit Tests	xUnit / MSTest	`src/modules/*/UnitTests/`	.NET module logic
UI Tests	WinAppDriver / Appium	`src/tests/`	Settings UI, FancyZones UI, Peek UI
Fuzz Tests	Custom harness	`src/modules/AdvancedPaste/`	Input validation for paste formats
Verification Scripts	PowerShell	`tools/Verification scripts/` (2 files, 37K chars)	Pre-release verification

Running tests:

```powershell
# Run all C# tests
dotnet test PowerToys.sln --configuration Release

# Run specific test project
dotnet test src/modules/fancyzones/tests/UnitTests/FancyZonesUnitTests.csproj

# Run C++ tests via vstest
vstest.console.exe src\modules\keyboardmanager\test\Release\KeyboardManagerTest.dll
```

> **Note**
>
> UI tests require a desktop session and cannot run in headless CI. The CI pipeline runs unit tests on every PR; UI tests run on a dedicated test machine nightly.

8.6 Key Architectural Patterns

Settings ViewModels (`src/settings-ui/Settings.UI.Library/`): Every module has a corresponding ViewModel that follows the MVVM pattern. Settings changes flow through the ViewModel, which serializes to JSON and writes to disk. The 300ms debounce prevents rapid file I/O during slider adjustments or other continuous inputs.

Module Registration (`src/common/utils/modulesRegistry.h`): Modules self-register via a static registry pattern. The Runner queries this registry at startup to discover available modules and their capabilities (hotkey support, elevated mode, preview handlers).

Interop Layer (`src/common/interop/`): Bridges the C++ Runner with .NET modules. This layer handles marshaling of strings, structs, and callbacks between native and managed code.

HTTP Client (`src/common/utils/HttpClient.h`): Used exclusively for update checking against GitHub's API. Respects system proxy settings and metered connection status.

Process Utilities (`src/common/utils/processApi.h`): Provides helpers for process enumeration, elevation detection, and inter-process signaling used throughout the Runner and modules.

Time Utilities (`src/common/utils/timeutil.h`): Standardized time handling for scheduling (Awake timers, update intervals, debounce windows).

8.7 Contributing a New Module

Development workflow for adding a new module:

1. Create the module project under `src/modules/{ModuleName}/`.
2. Implement the standard module interface (C++ DLL exports or .NET interface).
3. Define named event constants in `src/common/interop/Constants.cpp`.
4. Create a Settings ViewModel in `src/settings-ui/Settings.UI.Library/ViewModels/`.
5. Add a Settings page in `src/settings-ui/Settings.UI/Views/`.
6. Create an installer component (`.wxs` file) in `installer/`.
7. Add a DSC schema entry in the schema generator.
8. Write unit tests covering core logic.
9. Run the Verification Scripts before submitting a pull request.

9. Integration Points

9.1 Windows Shell Integration

PowerToys integrates deeply with the Windows shell:

Integration	Mechanism	Modules
Context menus	Shell extension (IContextMenu)	Image Resizer, Power Rename, File Locksmith, New+
Preview pane	Preview handler (IPreviewHandler)	File Explorer Add-ons (SVG, MD, PDF, G-code, etc.)
Thumbnail	Thumbnail provider (IThumbnailProvider)	File Explorer Add-ons
Task Scheduler	COM Task Scheduler API	Runner auto-start
System tray	Shell_NotifyIcon	Runner notification area icon
Global hotkeys	RegisterHotKey / WH_KEYBOARD_LL	All hotkey-enabled modules

9.2 PowerShell Integration

Command Not Found module:

```powershell
# Installed as a PowerShell module
Import-Module Microsoft.PowerToys.CommandNotFound

# Hooks into PowerShell's CommandNotFoundAction
```

```
# When a command is not found, queries WinGet for matching packages
# Example output:
# PS&gt; ffmpeg
# ffmpeg: The term 'ffmpeg' is not recognized...
#   Suggestion: winget install Gyan.FFmpeg
```

> **Caution**
>
> The Command Not Found module adds latency to every failed command lookup since it queries the WinGet index. On machines without internet connectivity, this can add significant delays to PowerShell startup and command execution (#33669). In environments that allow optional installation of WinGet components, the module may also attempt installation with missing prerequisites (#31091). Consider disabling it on air-gapped or restricted-network machines.

9.3 DSC v3 / WinGet Configuration

PowerToys settings can be managed as part of a broader Windows configuration using `winget configure`:

yaml

```yaml
# Example: Full workstation setup including PowerToys
properties:
  configurationVersion: 0.2.0
  resources:
    # Install PowerToys
    - resource: Microsoft.WinGet.DSC/WinGetPackage
      directives:
        description: Install PowerToys
      settings:
        id: Microsoft.PowerToys
        source: winget

    # Configure PowerToys modules
    - resource: Microsoft.PowerToys.Configure/PowerToysConfigure
      dependsOn:
        - WinGetPackage
      settings:
        GeneralSettings:
          Startup: true
          Theme: system
        FancyZones:
          Enabled: true
          FancyzonesShiftDrag: true
        PowerToysRun:
          Enabled: true
          SearchResultPreference: SearchResultPreference_MostRecentlyUsed
        KeyboardManager:
          Enabled: true
```

9.4 OpenAI / Semantic Kernel (Advanced Paste)

Advanced Paste uses Microsoft Semantic Kernel to integrate with OpenAI's API for AI-powered clipboard transformations. Configuration:

json

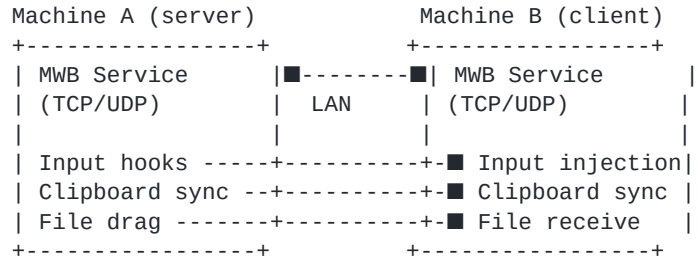
```
{
```

```
  "properties": {
    "OpenAIKey": "sk-...",
    "CustomActions": [
      {
        "Name": "Summarize",
        "Prompt": "Summarize the following text concisely:",
        "IsShown": true
      }
    ]
  }
}
```

The OpenAI key is stored in the module's local settings file. There is no central credential store integration. For enterprise environments, consider using a proxy/gateway that injects API keys rather than distributing keys to endpoints.

9.5 Multi-Machine (Mouse Without Borders)

Mouse Without Borders enables keyboard and mouse sharing across up to four Windows machines on the same network:



Key operational notes:

- Machines are paired via a one-time security key exchange
- Connection uses `MWB_RECONNECT_EVENT` to re-establish after network interruptions
- Clipboard synchronization is bidirectional
- File drag-and-drop transfers files over the LAN connection
- The `MWB_TOGGLE_EASY_MOUSE_EVENT` enables a simplified cursor transition mode

9.6 FancyZones CLI

FancyZones exposes a CLI interface for programmatic zone management:

```powershell
# Apply a layout via CLI (C# CLI component in src/modules/fancyzones/)
FancyZonesCLI.exe --apply-layout "{layout-guid}" --monitor 1
```

The CLI emits telemetry events (`FancyZonesCLICommandEvent`) with command name and success status, enabling automation visibility through ETW.

9.7 Telemetry Integration

For organizations that collect ETW telemetry centrally:

Provider: `Microsoft.PowerToys.Telemetry`

WPR capture workflow:

```powershell
# Start capture using the shipped WPR profile
wpr.exe -start "PowerToys.wprp"

# ... reproduce issue or collect steady-state data ...

# Stop and save
wpr.exe -stop "PowerToys_Trace.etl"

# Convert to human-readable format
tracerpt PowerToys_Trace.etl -o PowerToys_Events.xml -of XML
```

9.8 Diagnostic and Development Tools

Tool	Source Location	Purpose
BugReportTool	`tools/BugReportTool/` (17 files, 58K chars)	Collect logs, config, and system info for bug reports
CleanUp	`tools/CleanUp/`	Remove leftover files from failed or partial installs
MonitorReportTool	`tools/MonitorReportTool/` (7 files, 19K chars)	Monitor detection, resolution, and DPI diagnostics
StylesReportTool	`tools/StylesReportTool/` (5 files, 16K chars)	Window style flag reporting for overlay debugging
Module Loader	`tools/` (9 files, 69K chars)	Test individual module loading in isolation
FancyZones tools	`tools/`	Hit test, draw layout, and zonable tester utilities
MCP Server	`tools/` (Node.js, 4 files, 21K chars)	GitHub issue image/attachment handling via MCP
Verification Scripts	`tools/Verification scripts/` (2 files, 37K chars)	Pre-release verification checklists

> ***DocCompiler.ai*** *v3.1 | Upgraded source coverage | Generated 2026-05-07 | Source:* `microsoft/PowerToys` *@* `fde1599` *A Cyber Panda Solutions LLC product. This document is intentionally comprehensive -- built to be the complete, source-verified reference an AI or human needs without consulting other sources.* ***Do not print:*** *the always-current version lives at doccompiler.ai/microsoft/PowerToys.*