

# PatchWork

## Windows Update Manager



---

### Document Revision History

| Version | Date       | Notes         |
|---------|------------|---------------|
| 2.2.2   | 2026-05-14 | Point Release |

---

## TABLE OF CONTENTS

1. **Introduction**
2. **System Requirements and Prerequisites**
3. **Installation and Removal**
4. **Licensing and Registration**
5. **Concepts and Terminology**
6. **Quick Start**
7. **Command Reference**
8. **Output Formats and Reporting**
9. **Exit Codes**
10. **Deployment Scenarios**
11. **Automation and Scripting Patterns**
12. **Filtering Cookbook**
13. **Notifications**
14. **Security Considerations**
15. **Performance and Tuning**
16. **Troubleshooting**
17. **Comparison and Migration**
18. **Appendices**
19. **Support**

---

## CONVENTIONS USED IN THIS DOCUMENT

Command-line syntax follows these conventions throughout:

- **--switch** denotes a simple boolean flag that takes no value.
- **--switch VALUE** denotes a switch that requires an argument; **VALUE** names the argument type.
- **[OPTIONAL]** items may be omitted. The default behaviour is described in the relevant section.
- Code blocks show commands exactly as you would type them, using **patchwork** as the executable name regardless of your installation path.
- Admonitions are formatted as blockquotes prefixed with **Note:**, **Warning:**, or **Tip:**.

Registry paths use the abbreviated prefix **HKLM** for **HKEY\_LOCAL\_MACHINE**.

---

## 1. INTRODUCTION

PatchWork is a command-line tool for managing Windows Updates. It talks directly to the Windows Update Agent (WUA) via COM, which means there is no .NET runtime dependency, ensuring the binary is self-contained and compact (just a few megabytes in size). The tool covers the full update lifecycle: searching for available updates, downloading them, installing them, uninstalling them, and generating structured reports of what happened for each stage.

---

### WHY PATCHWORK?

The built-in Windows update management only cover part of the problem. **wuaclt.exe** and **Usoclient.exe** trigger background scans and installs but give no feedback and no filtering. **PSWindowsUpdate** is a capable PowerShell module but requires .NET and PowerShell execution policy consideration on some builds. PatchWork sits in the gap. It runs from a Windows console, script, via remote execution (PSEXec etc.), or via a scheduled task under SYSTEM. It is highly featured and accepts fine-grained filtering via the command line, and exits with a code that scripts can branch on.

---

### KEY FEATURES

- Full search, download, install, uninstall, and history operations for Windows Update, Microsoft Update, or via a WSUS server.
- Classification and severity filtering, KB number allow/deny lists, regex title matching, product filtering, size caps, release date windows, and update ID filtering — combinable in a single invocation.
- XML and JSON report output for downstream processing, SIEM ingestion, or compliance tooling.
- Email (SMTP) and syslog notifications after each run.
- Persistent default options can be stored in the registry for simplicity, with a per-run override capability.
- Pre- and post-operation custom actions (batch, PowerShell, or any other executable).
- Windows Update system health check that probes the environment to confirm the environment is working as required.

---

### INTENDED AUDIENCE

This document is aimed at Windows system administrators, MECM(SCCM)/Intune engineers, and anyone automating patch management via scripts, DevOps or scheduled tasks. A working knowledge of Windows Update concepts (WSUS, WUA, classifications, KB articles) is assumed throughout.

---

## 2. SYSTEM REQUIREMENTS AND PREREQUISITES

---

### OPERATING SYSTEM

PatchWork runs on Windows 7 and later, including all Windows Server editions from Server 2008 R2 onward. Both 32-bit and 64-bit platforms are supported. We will endeavour to support legacy systems with Extended Support, but official support is only provided for Windows versions that provide Active Support.

---

### PRIVILEGES

Most operations, such as download, install, uninstall, and anything that touches WSUS registry settings, **require administrative privileges**. Run PatchWork from an elevated Command Prompt, as a scheduled task under the SYSTEM account, or via **runas**. The **--search**, **--history**, **--installed**, and **--healthcheck** operations can run without elevation, though certain checks within **--healthcheck** will report reduced information if admin is unavailable.

---

### NETWORK

For Windows Update and Microsoft Update sources, outbound HTTPS to Microsoft's update endpoints must be reachable. For WSUS, the machine must be able to reach the WSUS server on its configured port (typically 8530 for HTTP, 8531 for HTTPS). SMTP and syslog notification features need outbound access to the configured mail or log server.

---

### DISK SPACE

There is no fixed disk space requirement for PatchWork itself. The binary is small. Update download and installation however, vary by patch content; use **--check-available-disk-space** to verify free space before a large run.

---

### 3. INSTALLATION AND REMOVAL

#### QUICK INSTALL

Run PatchWork once with **--setup** from an elevated prompt:

```
patchwork --setup
```

This copies the running executable to **C:\Program Files\Emerita\Patchwork\**, adds that directory to the System **PATH** environment variable, and writes an installation record to **HKLM\SOFTWARE\Emerita\Patchwork**. After setup, **patchwork** is available from any Command Prompt without specifying the full path (a new shell session, or a **refreshenv**, is needed for the PATH change to take effect in existing sessions).

#### CUSTOM INSTALL PATH

To install to a different directory, pass the desired path as an argument:

```
patchwork --setup "D:\Tools\PatchWork"
```

#### RECOMMENDED INSTALL

We would recommend you both register and provide default settings during install. You can chose a set of default options that work in most cases via **--opt-verbose**, but you can also provide custom setting that are tailored to your environment via **--opt-save**. A registered install can be as simple as :

```
patchwork.exe --setup --register "FirstName LastName/Companyname|XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX" --opt-verbose
```

To provide custom settings that suit your particular environment:

```
patchwork.exe --setup --register "FirstName LastName/Companyname|XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX" --opt-save --autoaccepteula --ignore-errors --logfile c:\windows\temp\patchwork.log --show-progress --color --info --hide-sensitive --wsus-server http://wsusserver.internal.pri
```

---

## RUNNING WITHOUT INSTALLING

PatchWork does not require installation. The executable can be placed anywhere on the file system and run directly. **--setup** is a convenience that handles the PATH registration; it is not a prerequisite.

---

## UPGRADING

Running **--setup** when PatchWork is already installed performs an upgrade rather than a fresh install. It compares the version of the running binary against the version recorded in the registry and, if the running binary is newer, copies it over the installed copy. The PATH and directory are left unchanged.

### **patchwork --setup**

To upgrade to a specific path, copy the new binary there and run with **--setup** pointing to the same directory used during the original install.

---

## UNINSTALLING

### **patchwork --remove**

This removes the installed executable, deletes the installation directory, removes the **PATH** entry, and deletes **HKLM\SOFTWARE\Emerita\Patchwork**. Administrator privileges are required.

---

## VERIFYING THE INSTALLATION

### **patchwork --version**

### **patchwork --healthcheck**

**--version** prints the version string. **--healthcheck** probes the environment more thoroughly to determine if the Windows Updater Agent API is working as expected (see [Section 7](#) for details).

---

## 4. LICENSING AND REGISTRATION

---

### INTERACTIVE REGISTRATION

#### **patchwork --register**

Prompts for a username and serial number, then stores the license. No other arguments are needed.

---

### UNATTENDED REGISTRATION

Pass the credentials on the command line, separated by a pipe character:

```
patchwork --register "FirstName LastName/Companyname|XXXXX-XXXXX-XXXXX-XXXXX"
```

The argument must be quoted if it contains spaces. This form is suitable for deployment scripts where interactive input is not available.

---

### COMBINING REGISTRATION WITH OTHER OPERATIONS

As mentioned above, **--register** can be combined with **--setup** and other configuration operations in a single invocation:

```
patchwork --setup --register "FirstName LastName/Companyname|XXXXX-XXX  
XX-XXXXX-XXXXX" --opt-save --use-windowsupdate
```

---

### LICENSE STORAGE

License data is managed by the Obsidium licensing system. If you receive a false positive from your AV solution please exclude the patchwork.exe from scanning, contact your AV vendor directly to ensure the executable is whitelisted, or contact us directly so we can contact the AV vendor on your behalf.

---

### UNLICENSED VERSIONS

If you are not a registered customer and are not using an evaluation version of PatchWork, it will report an unlicensed state on startup, will provide limited functionality and exit any operations with code 8 (**InvalidVersion**). No update operations are performed.

---

## 5. CONCEPTS AND TERMINOLOGY

### UPDATE SOURCES

PatchWork obtains updates from one of three sources, selected by the flags described in [Section 7](#).

**Windows Update (WU)** — Microsoft’s public update service for the Windows operating system. This is the default source when no WSUS server is detected in the registry.

**Microsoft Update (MU)** — A superset of Windows Update that also distributes updates for other Microsoft products such as Office. To use it, we recommend Microsoft Update service is first be registered with the local WUA instance (`--register-microsoftupdate`), or you can force it for a single run with `--use-microsoftupdate`.

**WSUS** — Windows Server Update Services, the enterprise update proxy. When WSUS is configured on a machine (via Group Policy, MECM(SCCM) or the registry), PatchWork will use it by default. You can override this with `--use-windowsupdate` or `--use-microsoftupdate`, or point to a specific/alternative server with `--wsus-server`.

### UPDATE CLASSIFICATIONS

The `--classification` switch accepts a string of single-letter codes, each enabling a category:

| Code | Classification | Typical content                              |
|------|----------------|--|
| C    | Critical       | Fixes for severe vulnerabilities and defects |
| U    | Security       | Security bulletins and vulnerability patches |
| D    | Definition     | Antivirus and antimalware signature updates  |
| I    | Update         | General improvements and non-security fixes  |
| R    | Rollup         | Cumulative rollup packages                   |
| S    | Service Pack   | Major packaged update collections            |
| F    | Feature Pack   | New feature additions                        |
| E    | Driver Sets    | Driver update collections                    |
| V    | Drivers        | Individual device driver updates             |
| G    | Upgrades       | Major OS version upgrades                    |

Codes can be combined. `--classification CU` limits results to Critical and Security updates. `--classification CUDISRF` covers most software updates while excluding drivers and upgrades.

### UPDATE SEVERITIES



The **--severity** switch accepts single-letter codes representing the MSRC severity rating assigned to an update:

| Code | Severity  | Meaning  |
|------|-----------|--|
| C    | Critical  | Exploitable remotely without user interaction                  |
| I    | Important | Could compromise system integrity or availability              |
| M    | Moderate  | Exploitability is mitigated by configuration or authentication |
| L    | Low       | Difficult to exploit; minimal impact                           |
| U    | Unknown   | No severity rating assigned                                    |

The **U** code is useful when filtering driver or definition updates, which often carry no severity value.

---

## THE SEARCH-DOWNLOAD-INSTALL LIFECYCLE

PatchWork separates the three stages of update deployment into discrete operations. Running **--install** performs all three internally (search, then download, then install). Running **--search** alone lets you review what is available before committing. Running **--download** stages updates to the local WUA cache without installing; a subsequent **--install** will use the cached copies.

This separation matters for scenarios where you want to pre-stage updates and then install during a separate maintenance window.

---

## DEFAULT OPTIONS AND PRECEDENCE

PatchWork supports a saved set of default options stored in the registry under **HKLM\Software\Emerita\Patchwork** as the **DefaultOptions** value. These are automatically prepended to the command line on every run.

**CLI arguments always take precedence over saved defaults.** If a saved default sets **--logfile C:\Logs\default.log** and the current command line specifies **--logfile C:\Logs\today.log**, the value from the command line wins.

Use **--opt-ignore** on any individual run to completely bypass the saved defaults for a particular run without deleting them.

---

## EXIT CODES

PatchWork exits with a numeric code that scripts can branch on. The full table is in [Section 9](#). The most commonly used codes are:

- **0** — success, no reboot needed
  - **1** — at least one error, no reboot needed
  - **3** — no updates matched the filter criteria
  - **10** — success, but one or more updates require a reboot
-

## 6. QUICK START

The following examples assume PatchWork is installed and the binary is in **PATH**. Run from an elevated Command Prompt unless noted otherwise.

**Check what updates are available:**

```
patchwork --search --info
```

**Search for Critical and Security updates only:**

```
patchwork --search --classification CU --info
```

**Download Critical and Security updates without installing:**

```
patchwork --download --classification CU
```

**Install Critical and Security updates, reboot automatically if required:**

```
patchwork --install --classification CU --autoaccepteula --reboot-if-needed
```

**Install all updates, log to file, no console output:**

```
patchwork --install --autoaccepteula --silent --logfile C:\Logs\updates.log
```

**Check the environment before running:**

```
patchwork --healthcheck
```

---

## 7. COMMAND REFERENCE

Options are grouped here as they appear in **--help**. One operation flag is required per invocation unless the command is a management-only action (**--register**, **--setup**, **--remove**, **--opt-save**, **--opt-clear**, **--opt-show**, **--opt-verbose**, **--list-exit-codes**, **--healthcheck**).

---

### 7.1 OPERATIONS

These flags determine what operation PatchWork performs. Exactly one primary operation (**--search**, **--download**, **--install**, **--uninstall**, **--history**, **--installed**) must be present per invocation, unless the operation is a management-only action listed below.

---

#### **--search**

Search queries the configured update source for available updates without downloading or installing anything. Apply any filter switches alongside **--search** to narrow the results. Combine with **--info** to provide full update details, or with **--xmlout/--jsonout** to produce an output report.

```
patchwork --search --classification CU --info
patchwork --search --releasedate days:30 --jsonout C:\Reports\pending.json
```

---

#### **--download**

Searches for updates matching the active filters and downloads them to the Windows Update local cache. The updates are ready to install on a subsequent **--install** run without downloading again.

```
patchwork --download --classification CU --severity CI
```

---

#### **--install**

Searches for matching updates, downloads any that are not already cached, and installs them. This is the most commonly used operation for a standard patching run.

**patchwork --install --classification CU --autoaccepteula --reboot-if-needed**

---

---

### **--uninstall**

Removes previously installed updates. Combine with **--kb**, **--match-filter**, or **--match-id** to target specific updates. Not all updates support uninstallation. PatchWork will report which ones are uninstallable before proceeding.

**patchwork --uninstall --kb KB5012345**

---

---

### **--history**

Lists the Windows Update installation history for the local machine. This is the equivalent of Settings → Windows Update → Update History. No filtering is applied; all recorded history is shown.

**patchwork --history --xmlout C:\Reports\history.xml**

---

---

### **--installed**

Lists updates currently installed on the system — the equivalent of Settings → Apps → Installed Updates. Combine with **--xmlout** or **--jsonout** for a structured inventory.

**patchwork --installed --jsonout C:\Reports\installed.json**

---

---

### **--register ["USER|KEY"]**

Registers the product license. Without arguments it will prompt interactively. With the **"USER|KEY"** argument, it registers unattended. See [Section 4](#).

---

---

### **--setup [PATH]**

Installs PatchWork within Windows adds it to the System PATH. Defaults to **C:\Program Files\Emerita\Patchwork** if no path is given. **Requires administrator privileges**. See [Section 3](#).

---

---

**--remove**

Uninstalls PatchWork. Removes the executable, directory, PATH entry, and registry keys. **Requires administrator privileges.**

---

---

## 7.2 UPDATE TYPE SELECTION

By default, PatchWork searches for software updates only. These switches change that scope.

---

---

**--driveronly**

Restricts the search to driver updates. Mutually exclusive with **--includedrivers**.

```
patchwork --search --driveronly --info
```

---

---

**--includedrivers**

Adds driver updates to the software update search. Mutually exclusive with **--driveronly**.

```
patchwork --install --includedrivers --classification CUV
```

---

---

**--alltypes**

Includes all update types that the WUA supports. Use this when you want to cast the widest possible net, for example when auditing a machine.

```
patchwork --search --alltypes --info
```

---

---

## **--preview**

Includes preview, optional, and beta updates, which are hidden from the default search. Use with care in production environments; preview updates are not yet fully validated. This options is not on versions of Window prior to Windows 10 1903 and Windows Server 2022. We attempt to work around issues using **--preview** on older builds, but we recommend in those cases, the switch is not used at all.

**patchwork --search --preview --info**

---



---

## 7.3 SEARCH CRITERIA AND FILTERING

These switches control which updates are included or excluded from an operation. When multiple filters are active, an update must satisfy all of them to be processed (AND logic). Within the **--kb** and **--match-id** switches, multiple values use OR logic.

---



---

## **--criteria CRITERIA**

Passes a raw WUA search criteria string directly to the Windows Update Agent. This overrides PatchWork's default criteria construction. Useful for edge cases not covered by the other filter switches.

Common criteria predicates:

| Predicate               | Meaning                                 |
|-------------------------|---|
| <b>IsInstalled=0</b>    | Not yet installed (PatchWork's default) |
| <b>IsInstalled=1</b>    | Already installed                       |
| <b>Type= 'Software'</b> | Software updates only                   |
| <b>Type= 'Driver'</b>   | Driver updates only                     |
| <b>IsHidden=0</b>       | Not hidden                              |

**patchwork --search --criteria "IsInstalled=0 AND Type='Software' AND IsHidden=0"**

---

---

## **--classification FLAGS**

Filters updates by classification category. Pass one or more letter codes as a single string without separators. See [Section 5](#) for the full code table.

```
patchwork --install --classification CU          # Critical and Security
patchwork --install --classification CUDISRF     # All software,
                                                no drivers
patchwork --search  --classification CUDISRFEVG  # Everything
```

---

---

## **--severity FLAGS**

Filters updates by MSRC severity rating. See [Section 5](#) for codes.

```
patchwork --install --severity CI    # Critical and Important only
patchwork --search  --severity CIML  # All rated updates
patchwork --search  --severity U     # Updates with no severity rating
```

---

---

## **--product PRODUCTS**

Includes only updates that belong to a matching product or category. The value is a comma-separated list of substrings; an update passes if any of its WUA category names contains any of the listed strings (case-insensitive substring match).

```
patchwork --search --product "Windows 10"
patchwork --search --product "Windows 11,Office"
```

---

---

## **--exclude-product PRODUCTS**

Excludes updates that match any of the listed product substrings. Follows the same matching logic as **--product**.

```
patchwork --install --exclude-product "Windows Defender"
patchwork --install --exclude-product "Office,Silverlight"
```

---



---

## **--kb KB\_NUMBERS**

Filters by KB article number. Accepts a comma-separated list. Prefix a KB number with - to exclude it; all others are treated as inclusions. The **KB** prefix is optional.

```
patchwork --install --kb KB5078740           # Include one KB
patchwork --install --kb KB5078740,KB5034441  # Include two KBs
patchwork --install --kb KB5078740,-KB5034441 # Include one,
                                                # exclude another
patchwork --search  --kb -KB5034441          # Exclude one KB,
                                                # show all others
```

When include entries are present, only those specific updates pass. When only exclude entries are present, everything except the excluded KBs passes.

---

## **--match-filter PATTERN**

Applies a regex pattern to update titles and descriptions. Only updates whose title or description matches the pattern are included. Standard .NET-compatible regex syntax applies.

```
patchwork --search --match-filter "Cumulative Update.*2025"
patchwork --search --match-filter "(?i)security"      # Case-insensitive
patchwork --search --match-filter "Windows (10|11)"
```

---

## **--nomatch-filter PATTERN**

Excludes updates whose title or description matches the regex pattern.

```
patchwork --install --nomatch-filter "Preview|Beta"
patchwork --install --nomatch-filter "Defender"
```

---

---

## **--matchfile FILE**

Loads include patterns from a text file instead of specifying them inline. One regex pattern per line. Blank lines and lines beginning with **#** are treated as comments and ignored. If the file contains multiple patterns, they are combined with OR logic (an update matching any pattern is included).

**Example file — critical-kbs.txt:**

```
KB5078740
KB5034441
Security Update.*2025
Cumulative Update for Windows
```

```
patchwork --install --matchfile C:\Config\critical-kbs.txt
```

**--matchfile** produces an include-only filter. Use **--nomatch-filter** for excludes.

---

## **--nomatchfile FILE**

Loads exclude patterns from a text file. The file format is identical to **--matchfile** — one regex per line, blank lines and **#** comments are ignored, multiple patterns combined with OR logic. An update matching any pattern in the file is excluded.

**Example file — excluded-kbs.txt:**

```
# Preview and beta releases
Preview
Beta
# Specific KBs to hold back
KB5034441
```

```
patchwork --install --nomatchfile C:\Config\excluded-kbs.txt
```

**--matchfile** and **--nomatchfile** can be combined with each other and with **--match-filter** / **--nomatch-filter** in the same run.

```
patchwork --install --matchfile C:\Config\approved.txt --nomatchfile C:\Config\excluded.txt
```

---

---

## **--releasedate DATE**

Filters updates by their release date. Several formats are accepted:

| Format               | Meaning  |
|----------------------|--|
| <b>YYYY-MM-DD</b>    | Released on or after this date (same as <b>ge:YYYY-MM-DD</b> ) |
| <b>ge:YYYY-MM-DD</b> | On or after (inclusive)  |
| <b>gt:YYYY-MM-DD</b> | Strictly after   |
| <b>le:YYYY-MM-DD</b> | On or before (inclusive)                                       |
| <b>lt:YYYY-MM-DD</b> | Strictly before  |
| <b>eq:YYYY-MM-DD</b> | Exact date match   |
| <b>days:N</b>        | Released within the last N calendar days                       |

```
patchwork --search --releasedate days:30
```

```
patchwork --search --releasedate ge:2025-01-01
```

```
patchwork --install --releasedate gt:2025-03-01
```

Updates with no release date recorded by WUA are always excluded when this filter is active.

---

## **--max-update-count COUNT**

Caps the number of updates that will be processed in a single run. Applied after all other filters, so COUNT reflects the final set. Useful for staged rollouts or when you want to limit the scope of a run. Outstanding updates will be picked up on subsequent runs.

```
patchwork --install --classification CU --max-update-count 10
```

---

## **--max-total-size size**

Caps the cumulative size of updates that will be processed. Size is evaluated in the order updates are returned; once adding the next update would exceed the cap, it and all subsequent updates are dropped. Accepts a numeric value with an optional suffix: **K**, **M**, **G**, or **T** (or two-letter variants **KB**, **MB**, **GB**, **TB**).

```
patchwork --download --max-total-size 500M
```

```
patchwork --download --max-total-size 2G
```

---

---

### **--match-id IDs**

Filters by update GUID. Accepts a comma-separated list of GUIDs. Prefix a GUID with - to exclude it. Matching is case-insensitive.

```
patchwork --install --match-id 9fb049d9-8ee3-4913-937f-196648006ca5
patchwork --install --match-id ID1,ID2,-ID3
```

---

---

### **--only-downloaded**

Restricts results to updates that have already been downloaded to the local WUA cache. Useful to run an install pass that uses only pre-staged content.

```
patchwork --install --only-downloaded --autoaccepteula
```

---

---

## 7.4 CONFIGURATION OPTIONS

These switches configure the update source and related service settings. Most write temporarily to the registry, overwriting any pre-defined values, and are restored when PatchWork exits. **--register-microsoftupdate** and **--clear-wsus-server** will make permanent changes however.

---

---

### **--register-microsoftupdate**

Registers the Microsoft Update service with the local Windows Update Agent, enabling updates for all Microsoft products (Office, Visio, etc.) in addition to Windows updates. This change persists after PatchWork exits. **Requires administrator privileges.**

```
patchwork --register-microsoftupdate
```

This only needs to be run once per machine.

---

---

## **--clear-wsus-server**

Removes the WSUS server configuration from the registry, causing the machine to fall back to Windows Update for subsequent update operations (by PatchWork or the OS). This is a permanent change. Requires administrator privileges.

**patchwork --clear-wsus-server**

---

---

## **--use-wsus**

Forces the use of WSUS as the update source, even if another source was saved as a default. This has no effect if WSUS is not already configured in the registry or if a WSUS server is not manually specified. It is primarily useful to restore WSUS as the source after a **--use-windowsupdate** or **--use-microsoftupdate** default has been saved.

---

---

## **--use-windowsupdate**

Bypasses WSUS and queries Windows Update directly. Applied for the duration of the current run only; the WSUS configuration in the registry is not modified.

**patchwork --search --use-windowsupdate**

---

---

## **--use-microsoftupdate**

Queries the Microsoft Update service directly, bypassing WSUS. Includes Office and other Microsoft product updates. Applied for the current run only.

**patchwork --search --use-microsoftupdate**

---

---

## **--wsus-server SERVER**

Temporarily points PatchWork at a specific WSUS server URL for the current run. The machine's existing WSUS registry configuration is restored on exit.

**patchwork --install --wsus-server http://wsus.corp.example.com:8530**  
**patchwork --install --wsus-server https://wsus.corp.example.com:8531**

---

---

**--use-mu-on-error**

If the WSUS server is unreachable, fall back to Microsoft Update for the current run.

---

---

**--use-wu-on-error**

If the WSUS server is unreachable, fall back to Windows Update for the current run.

---

---

**--targetgroup GROUP**

Sets the WSUS client-side target group for the current run. The group name must match one configured on the WSUS server. The registry is restored to its original state when PatchWork exits.

```
patchwork --install --targetgroup "Production_Servers"  
patchwork --install --wsus-server http://wsus.example.com:8530 --targetgroup "Pilot"
```

---

---

**--notargetgroup**

Removes the WSUS target group registry entries (**TargetGroup** and **TargetGroupEnabled**) before the operation, so the machine is treated as ungrouped for this run. The original values are restored on exit.

```
patchwork --search --notargetgroup
```

---

---

## 7.5 PROXY CONFIGURATION

Proxy settings are applied for the duration of the current run and restored on exit. All proxy switches affect how the WUA communicates with the update source. Currently WinHTTP proxies are supported. Socks proxies are NOT supported at this time.

---

---

**--disable-win-http-proxy**

Disables the WinHTTP proxy for this run.

---

---

**--disable-ie-proxy**

Disables the Internet Explorer proxy for this run.

---

---

**--auto-detect-proxy**

Enables WPAD (Web Proxy Auto-Discovery) via IE's AutoDetect setting.

---

---

**--proxy-address ADDRESS**

Specifies a proxy server address manually. Accepts a hostname, FQDN, or IP address.

```
patchwork --search --proxy-address proxy.corp.example.com --proxy-port 8080
```

---

---

**--proxy-port PORT**

Specifies the proxy server port. Requires **--proxy-address**.

---

## 7.6 REBOOT AND SHUTDOWN OPTIONS

At most, one of **--reboot**, **--reboot-if-needed**, **--shutdown**, or **--shutdown-if-needed** may be specified per invocation.

---

---

**--reboot**

Initiates a system reboot immediately after the operation completes, regardless of whether the installed updates require one. Returns exit code 5 on success, 6 on failure.

---

---

**--reboot-if-needed**

Initiates a reboot only if one or more installed updates report that a reboot is required. If no reboot is needed, PatchWork exits normally.

```
patchwork --install --classification CU --autoaccepteula --reboot-if-needed
```

---

---

**--shutdown**

Shuts the system down after the operation instead of rebooting.

---

---

**--shutdown-if-needed**

Shuts the system down if any installed update requires a reboot; exits normally otherwise.

---

---

**--force-close**

Forces applications to close before the reboot or shutdown proceeds. Use with care: applications will not have an opportunity to save data.

```
patchwork --install --reboot --force-close
```

---

---

**--delay SECONDS**

Pauses for the specified number of seconds before initiating a reboot or shutdown. Gives logged-in users time to save work when the reboot/shutdown is forced, rather than conditional.

```
patchwork --install --reboot-if-needed --delay 300 # 5-minute warnin  
g
```

---

---

**--reboot-message MESSAGE**

Displays the specified message in the Windows shutdown dialog before a reboot or shutdown.

```
patchwork --install --reboot-if-needed --delay 300 --reboot-message "R  
ebooting for monthly security updates in 5 minutes."
```

---

---

## 7.7 INSTALLATION OPTIONS

---



---

## **--autoaccepteula**

Automatically accepts End User License Agreements without prompting. Required for unattended operation.

```
patchwork --install --autoaccepteula
```

---

---

## **--force**

Forces re-download and re-installation of updates, including those already installed or already cached.

---

## **--ignore-errors**

Continues processing remaining updates if one download or install fails, rather than aborting the run. The exit code will still reflect that errors occurred. We recommend you use this option to **ensure that patching is not held up by individual transient failures.**

```
patchwork --install --ignore-errors --logfile C:\Logs\updates.log
```

---

---

## **--defender-fix**

If a Microsoft Defender Antivirus signature update fails during an --install run, automatically attempts a recovery by:

1. Removing the stale or corrupted definition files.
2. Triggering a fresh definition download and install.

This can resolve common situations where a Defender definition update is blocked by a previous failed or partial update, without requiring a manual intervention or a full system restart.

**Note:** --defender-fix is a **licensed feature** and requires a valid registered license. On unlicensed installations the recovery step is skipped and a message is logged instead.

**Note:** The recovery only triggers if the specific update titled "Security Intelligence Update for Microsoft Defender Antivirus" fails. Other update failures are unaffected.

```
patchwork --install --defender-fix
```

```
patchwork --install --classification CUD --defender-fix
```

---

---

## **--parallel-downloads N**

Sets the number of concurrent downloads. Accepts a value between 1 and 10. Default is 3. Higher values can improve throughput on fast connections but increase load on the WSUS server, network or proxy servers.

**Note:** --parallel-downloads is a **licensed feature** and requires a valid registered license. On unlicensed installations parallel downloads are fixed to 3 concurrent downloads.

```
patchwork --download --parallel-downloads 5
patchwork --download --parallel-downloads 1  # serialise for bandwidth
h-limited links
```

---

---

## 7.8 LOGGING AND REPORTING

---

---

### **--quiet**

Reduces console output to essential results only. Progress details, per-update listings, and informational banners are suppressed. Error messages and final counts are still shown.

---

---

### **--silent**

Suppresses all console output. Use for scheduled tasks where output is not captured. Pair with **--logfile** to preserve a record of what occurred.

```
patchwork --install --silent --logfile C:\Logs\nightly.log
```

---

---

### **--logfile FILE**

Writes all output to the specified file in addition to (or instead of, with **--silent**) the console.

```
patchwork --install --logfile "C:\Logs\patch-$(date /T).log"
```

---

---

### **--logmode MODE**

Controls log file behaviour when the file already exists. **overwrite** (default) truncates the file before writing. **append** adds to the existing content.

```
patchwork --install --logfile C:\Logs\updates.log --logmode append
```

---

---

### **--logencoding ENCODING**

Sets the character encoding for the log file. **Unicode** (default) writes UTF-16LE with CRLF line endings, which is readable in Notepad and most Windows tools. **ANSI** writes plain text with the system code page which may be useful for reading/processing with legacy tools.

```
patchwork --install --logfile C:\Logs\updates.log --logencoding ANSI
```

---

---

### **--xmlout FILE**

Writes a structured XML report to the specified path on completion. Can be combined with any operation. Compatible with **--list-exit-codes** and **--healthcheck** to produce machine-readable output from those commands.

```
patchwork --search --xmlout C:\Reports\scan.xml
```

---

---

### **--xmlout-with-bom**

Adds a UTF-8 BOM to the XML output file. Required for correct rendering in some spreadsheet applications (Excel, for example, uses the BOM to detect UTF-8 encoding).

```
patchwork --search --xmlout C:\Reports\scan.xml --xmlout-with-bom
```

---

---

### **--jsonout FILE**

Writes a structured JSON report to the specified path on completion.

```
patchwork --search --jsonout C:\Reports\scan.json
```

---

---

### **--info**

Prints detailed information about each update — title, KB article, classification, severity, size, release date, and description — to the console. Without **--info**, only a summary is shown.

```
patchwork --search --info
```

---

---

### **--show-progress**

Displays per-update download and installation progress on the console. Useful for interactive sessions. You may want to omit this switch for scheduled tasks.

---

---

## **--color**

Enables ANSI color output for errors and progress indicators. Requires a terminal that supports VT escape sequences (Windows Terminal, modern ConHost etc.).

---

---

## **--extended-error**

Changes exit code semantics to a bitmap combining multiple status flags. See [Section 9](#) for the full flag table.

---

---

## **--simple-error**

Collapses exit codes to **0** (success) or **1** (any error). Useful for integration with systems that expect only a pass/fail return.

---

---

## **--debug**

Enables structured debug tracing. Writes a full JSON span log to **%TEMP%\patchwork-debug-  
<pid>.log** and mirrors DEBUG-level events to stderr. Also activates per-event forwarding to syslog and email sinks when those are configured. This output is primarily for diagnostics and support bundle generation. Typically the only time you should require this switch is if the output is required by Emerita support.

---

---

## **--hide-sensitive**

Redacts sensitive values in console output and log files. Specifically, the values passed to **--smtp-user**, **--smtp-password**, and **--register** are replaced with **\*\*\*\*\***. Use when log files may be reviewed by third parties or forwarded to monitoring systems.

---

---

## **--list-exit-codes**

Prints a table of all exit codes and their meanings, then exits. Combine with **--xmlout** or **--jsonout** to write the table as a report.

```
patchwork --list-exit-codes
```

```
patchwork --list-exit-codes --jsonout C:\Reports\codes.json
```

---

---

## **--healthcheck**

Runs a series of environment checks and reports the result for each. Exits with code 0 if all checks pass, code 1 if any check fails. Checks include: administrator privilege status, WUA service availability, COM class registration, WSUS connectivity (if configured), disk space (warn at < 500 MB free, fail at < 100 MB free), pending reboot state, proxy configuration, and recent update timestamps.

Combine with **--xmlout** or **--jsonout** for a machine-readable report suitable for monitoring pipelines.

```
patchwork --healthcheck
```

```
patchwork --healthcheck --xmlout C:\Reports\health.xml
```

---

---

## 7.9 TIMEOUT AND RUNTIME

---

---

### **--maxruntime SECONDS**

Sets a hard upper limit on total execution time. If the limit is exceeded before the operation completes, PatchWork exits with code 12 (**TimeoutReached**).

```
patchwork --install --maxruntime 3600 # Allow up to 1 hour
```

---

---

### **--retrycount COUNT**

Number of retry attempts for failed search, download, or install operations. Retries use an exponential backoff with jitter: initial delay 2 seconds, maximum delay 30 seconds.

```
patchwork --install --retrycount 5
```

---

---

## **--noretry**

Disables automatic retry entirely. PatchWork will fail immediately on the first error.

---

---

## 7.10 CUSTOM ACTIONS

Custom actions run synchronously, under the same account token as PatchWork itself. The working directory is inherited from the parent process.

### **Dispatch logic:**

- If the command's first token ends in **.ps1**, the script is run as: **powershell.exe - ExecutionPolicy Bypass -NonInteractive -File <path> [args]**
- All other commands (executables, **.cmd**, **.bat**, and shell built-ins) are run as: **cmd.exe /C <command>**

If a custom action exits with a non-zero code, PatchWork logs the failure but continues — it does not abort the operation. Check the log file to confirm custom actions completed successfully.

---

---

## **--custom-action-before COMMAND**

Runs a command before the main operation begins. Use to stop services, snapshot VMs, or check preconditions.

```
patchwork --install --custom-action-before "net stop MyAppService"
patchwork --install --custom-action-before "C:\Scripts\pre-patch.ps1"
patchwork --install --custom-action-before "net stop Svc1 & net stop Svc2"
```

To chain multiple commands: use **&** within a quoted string, or wrap in a **.cmd** file.

---

---

## **--custom-action-after COMMAND**

Runs a command after the main operation completes, regardless of outcome.

```
patchwork --install --custom-action-after "net start MyAppService"
patchwork --install --custom-action-after "C:\Scripts\post-patch.ps1 - SendReport"
```

---

---

## 7.11 SYSTEM CHECKS

---

---

### **--check-available-disk-space DRIVE[:SIZE]**

---

Reports the free space on the specified drive. Accepts the drive letter with or without a colon or backslash (**C**, **C:**, **C:\** are all equivalent; only the first character is used).

An optional minimum free-space requirement can be appended directly after the drive letter. The size format is the same as **--max-total-size** a number followed by an optional suffix (**K**, **KB**, **M**, **MB**, **G**, **GB**, **T**, **TB**), or a plain number for bytes. If the available free space is less than the specified size, PatchWork logs an error and exits immediately, before carrying out any other operation.

When combined with a primary operation (**--search**, **--download**, **--install**, etc.), the space check is reported first and then the operation continues. When used alone, PatchWork exits after reporting.

```
patchwork --check-available-disk-space C:
```

```
patchwork --check-available-disk-space C:10G
```

```
patchwork --install --check-available-disk-space C: --classification U
```

---

---

### **--refresh-last-update-timestamps**

---

Writes the current date and time to the Windows Update timestamp registry entries (**LastSearchTime**, **LastDownloadTime**, **LastInstallTime**, **LastUninstallTime**, **LastCheckTime**). This is occasionally needed to correct misleading “last checked” dates or to unify dates.

```
patchwork --refresh-last-update-timestamps
```

---

---

## 7.12 DEFAULT OPTIONS MANAGEMENT

---

Default options are stored as a string in the registry at **HKLM\Software\Emerita\Patchwork** under the value **DefaultOptions**. On every run, PatchWork reads this string, prepends it to the actual command line, and parses the combined result. Explicitly supplied command-line arguments always override saved defaults. Certain options, such as reboot/shutdown related switches, default operations and registration/setup related switches will not be saved. If you do require a switch that PatchWork refuses to save, then you can edit the registry manually that may overcome the limitation you are encountering. We may not be able to support you in these instances though.





---

## **--opt-save**

Saves the saveable options from the current command line as the new persistent defaults. Not all switches are saved — operation flags (**--search**, **--install**, etc.), reboot flags, and the management flags themselves are excluded. The intent is to save configuration options (source selection, logging, filtering preferences), not one-time actions.

**Requires administrator privileges** (writes to HKLM).

When combined with a primary operation, **--opt-save** saves first and then runs the operation.

```
patchwork --opt-save --use-windowsupdate --autoaccepteula --logfile C:\Logs\patchwork.log
```

---

---

## **--opt-clear**

Removes the saved defaults from the registry.

```
patchwork --opt-clear
```

---

---

## **--opt-show**

Prints the currently saved default options string without running any operation.

```
patchwork --opt-show
```

---

---

## **--opt-ignore**

Skips loading the saved defaults for this run. The registry value is not modified; it remains in place for subsequent runs.

```
patchwork --install --opt-ignore --use-windowsupdate
```

---

---

## **--opt-verbose**

Applies a preset that enables the following useful options by default without requiring a lot of additional effort to determine and understand what might be appropriate:

**--autoaccepteula --ignore-errors --logfile %TEMP%\patchwork.log --show-progress --color --info --hide-sensitive**

Any of these can be individually overridden on the same command line.

When used without a primary operation, **--opt-verbose** saves the preset as the default options (equivalent to running **--opt-save** with those flags). When combined with a primary operation, the preset is applied for that run without being saved.

```
patchwork --opt-verbose --install --classification CU    # Run with ver
bose preset
patchwork --opt-verbose                                # Save verbos
e preset as defaults
patchwork --opt-verbose --install --logfile D:\log.log  # Verbose, bu
t override logfile
```

---

## 8. OUTPUT FORMATS AND REPORTING

### CONSOLE OUTPUT

The console output level is controlled by the output mode switches:

| Mode   | Switch          | Description  |
|--------|-----------------|--|
| Normal | (none)          | Standard summaries and results; no per-update detail   |
| Info   | <b>--info</b>   | Full additional per-update detail including descriptions   |
| Quiet  | <b>--quiet</b>  | Counts and errors only; suppresses progress and banners  |
| Silent | <b>--silent</b> | No console output at all   |
| Color  | <b>-color</b>   | Adds ANSI color to output. Including errors (red), success (green) and progress indicators (blue/orange). It ensure output is significantly easier to interpret for humans. It has no effect in silent mode. |

### LOG FILES

Log files capture the same content as the console at the selected verbosity level. The encoding defaults to UTF-16LE (**Unicode**); use **--logencoding ANSI** for plain text. The mode defaults to **overwrite**; use **--logmode append** to accumulate across runs.

### XML OUTPUT

The XML report generated by **--xmlout** contains a root **<PatchWorkReport>** element with a **<Summary>** section and an **<Updates>** collection. Each **<Update>** element includes:

- **<Title>** — update title
- **<KBArticleID>** — KB number
- **<Classification>** — update classification
- **<Severity>** — MSRC severity
- **<Size>** — download size in bytes
- **<ReleaseDate>** — YYYY-MM-DD
- **<Description>** — full update description
- **<UpdateID>** — WUA GUID

Add **--xmlout-with-bom** to prefix the file with a UTF-8 BOM for compatibility with Excel and similar tools.

---

## JSON OUTPUT

The JSON report generated by **--jsonout** follows the same logical structure as the XML output, with a top-level **summary** object and an **updates** array.

---

## DEBUG TRACE

**--debug** produces a structured JSON span log at **%TEMP%\patchwork-debug-<pid>.log**. The file contains timestamped event records covering every major operation — WUA calls, filter decisions, download progress, and error details. This file is intended for support and diagnostics; send it alongside the regular log file and any potential .dmp file when reporting an issue.

---

## 9. EXIT CODES

### STANDARD EXIT CODES

| Code | Name                    | Meaning  |
|------|-------------------------|--|
| 0    | Success                 | Operation completed; no reboot required                    |
| 1    | ErrorNoReboot           | One or more errors occurred; no reboot required            |
| 2    | NoMoreUpdates           | No further updates are available                           |
| 3    | NoUpdatesMatchingFilter | No updates matched the active filter criteria              |
| 4    | InvalidCriteria         | The WUA search criteria were rejected as invalid           |
| 5    | RebootSuccess           | Reboot or shutdown initiated successfully                  |
| 6    | RebootFailed            | Reboot or shutdown could not be initiated                  |
| 7    | SyntaxError             | A command-line argument was invalid or missing             |
| 8    | InvalidVersion          | The product is unlicensed or the license has expired       |
| 10   | SuccessRebootRequired   | Operation completed; at least one update requires a reboot |
| 11   | ErrorWithReboot         | One or more errors occurred and a reboot is also required  |
| 12   | TimeoutReached          | The <code>--maxruntime</code> limit was exceeded           |

---

## EXTENDED EXIT CODES (-EXTENDED-ERROR)

When **--extended-error** is active, the exit code is a bitmap combining the following flags:

| Bit | Hex mask | Meaning  |
|-----|----------|--|
| 0   | 0x001    | A Windows Update error occurred                        |
| 1   | 0x002    | More updates match the filter than were processed      |
| 2   | 0x004    | More updates are available overall (beyond the filter) |
| 3   | 0x008    | The <b>--max-update-count</b> limit was reached        |
| 4   | 0x010    | A reboot is required                                   |
| 5   | 0x020    | The timeout limit was reached                          |
| 6   | 0x040    | Invalid search criteria                                |
| 7   | 0x080    | Syntax error   |
| 8   | 0x100    | Invalid license or version                             |
| 9   | 0x200    | Insufficient disk space                                |

For example, an exit code of **26 (0x1A = 0b00011010)** indicates: more updates matching filter available (bit 1), max update count reached (bit 3), and reboot required (bit 4).

### Parsing in PowerShell:

```
$code = $LASTEXITCODE
if ($code -band 0x10) { Write-Host "Reboot required" }
if ($code -band 0x08) { Write-Host "max-update-count was hit; more updates may remain" }
if ($code -band 0x01) { Write-Host "A Windows Update error occurred" }
```

### Parsing in CMD:

```
patchwork --install --extended-error --max-update-count 5 --classification CU
set /a REBOOT_REQ=%ERRORLEVEL% ^& 16
if %REBOOT_REQ% GTR 0 echo Reboot required
```

---

## SIMPLE ERROR MODE (-SIMPLE-ERROR)

All non-zero exit codes collapse to **1**. Use when the calling script only needs to know success or failure.

---

## USING EXIT CODES IN SCRIPTS

A typical pattern in a batch file:

```
patchwork --install --classification CU --autoaccepteula --reboot-if-needed
if %ERRORLEVEL% EQU 0 echo All done, no reboot needed.
if %ERRORLEVEL% EQU 10 echo Install succeeded - rebooting now.
if %ERRORLEVEL% EQU 3 echo No matching updates found.
if %ERRORLEVEL% EQU 1 echo Install completed with errors.
if %ERRORLEVEL% EQU 12 echo Timed out before all updates were installed.
```

---



## 10. DEPLOYMENT SCENARIOS

---

### STANDALONE WORKSTATION USING WINDOWS UPDATE

The simplest case: no WSUS in play, direct Windows Update access.

```
patchwork --install --classification CU --severity CI ^  
  --autoaccepteula --reboot-if-needed --delay 60 ^  
  --logfile "C:\Logs\patch-%DATE:~10,4%%DATE:~4,2%%DATE:~7,2%.log"
```

Run this from a scheduled task under SYSTEM, daily or weekly, during off-hours.

---

### DOMAIN-JOINED CLIENT USING WSUS

If the machine is already Group Policy-targeted at a WSUS server, PatchWork will use it automatically. To specify the target group explicitly:

```
patchwork --install --classification CU --autoaccepteula ^  
  --targetgroup "Production_Desktops" ^  
  --reboot-if-needed --logfile C:\Logs\update.log
```

If the WSUS server is temporarily unreachable, add **--use-wu-on-error** to fall back to Windows Update rather than failing the run outright.

---

### WSUS FALLBACK ON SERVER OUTAGE

```
patchwork --install --classification CU --autoaccepteula --use-wu-on-e  
rror ^  
  --logfile C:\Logs\update.log --xmlout C:\Reports\update.xml
```

---

### SERVER CORE AND HEADLESS DEPLOYMENTS

Headless systems have no interactive session. Run PatchWork silently, log to file, and parse the exit code in the calling script:

```
patchwork --install --classification CU --autoaccepteula ^  
  --silent --logfile C:\Logs\update.log ^  
  --xmlout C:\Reports\update.xml ^  
  --reboot-if-needed  
if %ERRORLEVEL% EQU 10 shutdown /r /t 300
```

---

### RUNNING UNDER SYSTEM VIA TASK SCHEDULER

Create a scheduled task with:

- **Action:** `patchwork.exe --install --classification CU --autoaccepteula --silent --logfile C:\Logs\patch.log --reboot-if-needed`
- **Run as:** SYSTEM
- **Run with highest privileges:** Yes
- **Trigger:** Weekly, outside business hours

No interactive login is needed. `--silent` suppresses any attempted console output.

---

## SCCM/MECM PACKAGE OR SCRIPT DEPLOYMENT

Deploy as a package or script step. The exit code maps to MECM's success/failure reporting:

```
patchwork --install --classification CU --autoaccepteula --ignore-errors ^
--quiet --logfile "%TEMP%\patchwork-mecm.log"
exit /b %ERRORLEVEL%
```

MECM treats exit code 0 as success and any other code as failure. If updates require a reboot (exit code 10), configure the deployment to handle a soft reboot.

---

## INTUNE WIN32 APP DEPLOYMENT

Package PatchWork as a Win32 app. Set the install command and define custom return codes in Intune:

- **Install command:** `patchwork.exe --install --classification CU --autoaccepteula --ignore-errors --quiet`
- **Return codes:**
  - **0** — Success
  - **10** — Success with reboot required (map to Intune's "soft reboot" code 3010)
  - **3** — No updates found (map to Success)
  - **1** — Failure

---

## STAGED PILOT ROLLOUT

Download updates on a representative pilot machine, validate, then deploy to production:

**rem Phase 1: Download on pilot machine**

```
patchwork --download --classification CU --logfile C:\Logs\pilot-downl
oad.log
```

```
xcopy C:\Windows\SoftwareDistribution\Download \\fileserver\Updates\Mo
nthly /E /I
```

**rem Phase 2: Install on pilot**

```
patchwork --install --classification CU --autoaccepteula ^
--logfile C:\Logs\pilot-install.log --xmlout C:\Reports\pilot.xml ^
--reboot-if-needed
```

**rem Phase 3: After validation, deploy broadly**

---

## CITRIX AND RDS GOLD IMAGE PATCHING

Patch the gold image before sealing. On a snapshot-based workflow:

1. Boot the gold image.
2. Run PatchWork against Windows Update or an approved WSUS group.
3. Verify the result via **--xmlout**.
4. Reboot if required.
5. Repeat until **--search** returns code 3 (no updates remaining).
6. Seal and publish the image.

```
:loop
```

```
patchwork --install --classification CU --autoaccepteula --quiet ^
--xmlout C:\Temp\patch-result.xml
```

```
if %ERRORLEVEL% EQU 10 (
    shutdown /r /t 0
```

```
)
```

```
if %ERRORLEVEL% EQU 0 goto done
```

```
if %ERRORLEVEL% EQU 3 goto done
```

```
echo Errors during patching - review C:\Temp\patch-result.xml
```

```
:done
```

---

## 11. AUTOMATION AND SCRIPTING PATTERNS

### PARSING JSON OUTPUT IN POWERSHELL

```
patchwork --search --classification CU --jsonout "$env:TEMP\scan.json"
| Out-Null
$report = Get-Content "$env:TEMP\scan.json" | ConvertFrom-Json

foreach ($update in $report.updates) {
    Write-Host "$($update.title) - $($update.kbArticleId) - $($update.
size) bytes"
}

Write-Host "Total: $($report.summary.totalUpdates) updates"
```

### HANDLING EXIT CODES IN POWERSHELL

```
patchwork --install --classification CU --autoaccepteula --reboot-if-n
eeded
switch ($LASTEXITCODE) {
    0 { Write-Host "All updates installed. No reboot needed." }
    10 { Write-Host "Updates installed. Rebooting in 5 minutes."; Star
t-Sleep 300; Restart-Computer -Force }
    3 { Write-Host "No updates matching filter." }
    1 { Write-Error "Install completed with one or more errors." }
    12 { Write-Error "Timed out. Some updates may not have been instal
led." }
    default { Write-Error "Unexpected exit code: $LASTEXITCODE" }
}
```

### SAVING SITE-WIDE DEFAULTS

To configure a standard set of options that apply to every PatchWork invocation on a machine without repeating them on every command line, save them as defaults. Run once from an elevated prompt:

```
patchwork --opt-save --use-windowsupdate --autoaccepteula ^
--logfile C:\Logs\patchwork.log --logmode append ^
--xmlout C:\Reports\patchwork.xml --hide-sensitive
```

From that point on, a simple **patchwork --install --classification CU** will automatically include all those options. Override any saved default by specifying it explicitly on the command line.

To check what is currently saved:

```
patchwork --opt-show
```

To clear everything:

```
patchwork --opt-clear
```

---

## PRE/POST CUSTOM ACTIONS FOR SERVICE CONTROL

Stop dependent services before patching, restart them after:

```
patchwork --install --classification CU ^  
  --custom-action-before "net stop MyAppService & net stop MyDBService  
" ^  
  --custom-action-after "net start MyDBService & net start MyAppServi  
ce" ^  
  --autoaccepteula --logfile C:\Logs\patch.log
```

For more complex pre- and post-logic, wrap in scripts:

```
patchwork --install --classification CU ^  
  --custom-action-before "C:\Scripts\pre-patch.ps1" ^  
  --custom-action-after "C:\Scripts\post-patch.ps1" ^  
  --autoaccepteula
```

**pre-patch.ps1** For example, could be used to snapshot a VM or drain a load balancer. **post-patch.ps1** might run smoke tests or send a Teams notification.

---

## IDEMPOTENT REBOOT HANDLING

You can easily check for or deploy patches via a scheduler that runs on every startup (e.g., a Group Policy startup script), or a simple Task Scheduler run on login:

```
patchwork --search --classification CU --severity CI
if %ERRORLEVEL% EQU 3 (
    echo No updates pending. Done.
    exit /b 0
)
```

```
patchwork --install --classification CU --severity CI ^
--autoaccepteula --reboot-if-needed ^
--logfile C:\Logs\startup-patch.log
```

---

## BANDING PILOT RINGS WITH COUNT AND DATE LIMITS

Install only the oldest N updates first, moving to newer ones in subsequent passes:

```
rem Ring 1: install up to 5 updates released more than 30 days ago
patchwork --install --classification CU ^
--releasedate le:2025-03-01 --max-update-count 5 ^
--autoaccepteula --logfile C:\Logs\ring1.log
```

---

## 12. FILTERING EXAMPLES

---

SECURITY UPDATES ONLY, LAST 30 DAYS

```
patchwork --search --classification U --releasedate days:30 --info
```

---

CRITICAL AND SECURITY UPDATES, CRITICAL AND IMPORTANT SEVERITY

```
patchwork --install --classification CU --severity CI --autoaccepteula
```

---

EVERYTHING EXCEPT A SPECIFIC KB

```
patchwork --install --kb -KB5034441
```

---

ALL SOFTWARE UPDATES EXCEPT DEFINITIONS

```
patchwork --install --classification CUISRF --autoaccepteula
```

---

CUMULATIVE UPDATES BY REGEX

```
patchwork --search --match-filter "Cumulative Update for Windows" --info
```

---

DRIVER UPDATES FROM A SPECIFIC VENDOR

```
patchwork --search --driveronly --product "Intel" --info
patchwork --install --driveronly --product "NVIDIA" --autoaccepteula
```

---

EXCLUDE DRIVERS FROM A BROAD INSTALL

```
patchwork --install --classification CUISRF --exclude-product "Realtek,
Broadcom"
```

---

LOAD A CURATED KB ALLOW-LIST FROM A FILE

Create **approved-kbs.txt**:

```
# Monthly approved patches - approved 2025-05-01
KB5078740
KB5034441
KB5036893
```

```
patchwork --install --matchfile C:\Config\approved-kbs.txt --autoaccepteula
```

---

## COMBINING CLASSIFICATION, SEVERITY, REGEX, AND DATE

```
patchwork --install ^  
  --classification CU ^  
  --severity CI ^  
  --match-filter "Windows (10|11|Server 2022)" ^  
  --releasedate ge:2025-01-01 ^  
  --nomatch-filter "Preview" ^  
  --max-update-count 20 ^  
  --autoaccepteula
```

---

## HOW FILTERS INTERACT

Filters are applied in this order:

1. WUA query (classification scope from update type switches, default **IsInstalled=0**)
2. Classification filter (**--classification**)
3. Severity filter (**--severity**)
4. KB include/exclude list (**--kb**)
5. Update ID include/exclude list (**--match-id**)
6. Regex include (**--match-filter** or **--matchfile**)
7. Regex exclude (**--nomatch-filter** or **--nomatchfile**)
8. Product include (**--product**)
9. Product exclude (**--exclude-product**)
10. Only-downloaded filter (**--only-downloaded**)
11. Preview filter (excluded unless **--preview**)
12. Release date filter (**--releasedate**)
13. Size cap (**--max-total-size**) — applied cumulatively, in update order
14. Count cap (**--max-update-count**) — truncates the final list

An update must pass all active filters. If no filters are specified for a given dimension, that dimension is not filtered (all values pass).

---



## 13. NOTIFICATIONS

### EMAIL NOTIFICATIONS

PatchWork can send an email report after any primary operation. The minimal required configuration is **--smtp-server**, **--email-from**, at least one **--email-to**, and **--send-email-on-completion** which is required to ensure the email send will be triggered.

### PORT AND ENCRYPTION MATRIX

| Port | Encryption switch value | Protocol           |
|------|-------------------------|--------------------|
| 25   | <b>none</b>             | Plain SMTP         |
| 587  | <b>starttls</b>         | SMTP with STARTTLS |
| 465  | <b>sslts</b>            | SMTP over SSL/TLS  |

### AUTHENTICATED SMTP

```
patchwork --install --classification CU ^
--smtp-server smtp.corp.example.com --smtp-port 587 ^
--smtp-encryption starttls ^
--smtp-user patchwork@corp.example.com ^
--smtp-password "secretpassword" ^
--email-from patchwork@corp.example.com ^
--email-to sysadmin@corp.example.com ^
--email-subject "Patch Run Complete - %COMPUTERNAME%" ^
--send-email-on-completion ^
--hide-sensitive
```

**--hide-sensitive** prevents the SMTP password from appearing in the log file and being output to the display.

### MULTIPLE RECIPIENTS

Specify **--email-to** more than once:

```
patchwork --install ... ^
--email-to alice@example.com ^
--email-to bob@example.com ^
--send-email-on-completion
```

---

## EMAIL SUBJECT

If `--email-subject` is not specified, PatchWork uses **PatchWork <operation> Report - <status>** (e.g. **PatchWork install Report - Success**).

---

## SAVING EMAIL CONFIGURATION AS DEFAULTS

```
patchwork --opt-save ^
  --smtp-server smtp.corp.example.com ^
  --smtp-port 587 --smtp-encryption starttls ^
  --smtp-user svc-patchwork@corp.example.com ^
  --smtp-password "password" ^
  --email-from svc-patchwork@corp.example.com ^
  --email-to ops-team@corp.example.com ^
  --send-email-on-completion --hide-sensitive
```

Once saved, every subsequent **patchwork --install** run will send a notification without extra arguments.

---

## SYSLOG NOTIFICATIONS

PatchWork sends a single RFC-5424 syslog message after each operation. The default transport is UDP.

---

## BASIC SYSLOG SETUP

```
patchwork --install --classification CU ^
  --syslog-server siem.corp.example.com ^
  --syslog-port 514 ^
  --syslog-protocol udp ^
  --syslog-facility local0 ^
  --syslog-tag patchwork ^
  --send-syslog-on-completion
```

---

## TCP VS UDP

UDP is the default and is appropriate for most internal networks. Use `--syslog-protocol tcp` if your SIEM requires reliable delivery or if the syslog server is across a WAN link.

```
patchwork --install ... --syslog-protocol tcp --syslog-server logs.example.com
```

---

## SYSLOG FACILITY

PatchWork supports **local0** through **local7**. Choose whichever facility your syslog server routes to the correct destination. The default is **local0**.

---

## SEVERITY MAPPING

| Operation outcome | Syslog severity sent |
|-------------------|----------------------|
| Success           | Notice               |
| Failed            | Warning              |
| Other             | Info                 |

---

## SAVING SYSLOG CONFIGURATION AS DEFAULTS

```
patchwork --opt-save ^  
  --syslog-server siem.corp.example.com ^  
  --syslog-port 514 --syslog-protocol udp ^  
  --syslog-facility local1 --syslog-tag patchwork ^  
  --send-syslog-on-completion
```

---

## 14. SECURITY CONSIDERATIONS

---

### PRIVILEGE MODEL

PatchWork follows the principle of least privilege where possible. **--search**, **--history**, **--installed**, and **--healthcheck** do not require elevation. All operations that write to the registry or modify the system (download, install, uninstall, setup, remove, target group changes, WSUS configuration) **require administrator privileges**. PatchWork checks for elevation and exits with an informative error if admin is required but not available.

---

### CREDENTIAL HANDLING

- **SMTP passwords** passed via **--smtp-password** appear in the command line and may be captured in process listings or audit logs. Use **--hide-sensitive** to redact them in PatchWork's own log output. Consider storing the entire SMTP configuration as saved defaults via **--opt-save** so the password does not appear in task scheduler or wrapper script command lines. The saved value is stored in the registry under **HKLM\Software\Emerita\Patchwork** so do not assume security. Ensure that SMTP account has minimal permissions to allow the sending of reports you require, but no more.
- **License keys** passed via **--register** are similarly sensitive. **--hide-sensitive** redacts the **--register** argument in logs.
- **Proxy credentials**: if proxy authentication is required, the password is not captured in PatchWork's log files even without **--hide-sensitive**, but be aware that it may appear in process audit logs.

---

### WHAT --HIDE-SENSITIVE COVERS

When **--hide-sensitive** is active, the values for the following switches are replaced with **\*\*\*\*\*** in all console output and log files:

- **--smtp-user**
- **--smtp-password**
- **--register**

---

## CODE SIGNING

The PatchWork executable is digitally signed with an Authenticode certificate issued to **Chad Matthieson**. Please verify the signature is present and valid before deploying, especially in sensitive environments. If the signature is not present or invalid, please contact us so we can investigate further. PatchWork contains code to ensure that the digital signature is present and correct prior to operation execution, so in the event of PatchWork not launching correctly, please redownload and replace the problematic executable.

**Get-AuthenticodeSignature "C:\Program Files\Emerita\Patchwork\patchwork.exe"**

---

## REGISTRY KEYS

PatchWork writes to and reads from the following registry locations:

| Location   | Purpose  |
|--|--|
| <b>HKLM\SOFTWARE\Emerita\Patchwork</b>                                   | Installation record ( <b>Installed, Version</b> ), default options ( <b>DefaultOptions</b> ), registration information |
| <b>HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Environment</b> | System PATH (written during <b>--setup</b> )   |
| WSUS client keys (temporary)   | Applied during operations with <b>--wsus-server</b> or <b>--targetgroup</b> ; restored on exit                         |
| WUA proxy configuration (temporary)                                      | Applied during proxy-switching operations; restored on exit  |

---

## HARDENING ON SHARED HOSTS

On jump servers or shared administrative hosts, **restrict who can invoke PatchWork** and who can write to **HKLM\Software\Emerita\Patchwork**. A low-privilege user who can write to **DefaultOptions** could inject flags (such as **--custom-action-before**) that **execute code with elevated privileges** on the next scheduled run.

---

## 15. PERFORMANCE AND TUNING

### PARALLEL DOWNLOADS

The **--parallel-downloads** switch controls how many updates are downloaded concurrently. The default of 3 is a reasonable middle ground for most environments. On fast LAN connections with a capable WSUS server, values up to 6 or 8 may improve throughput. On metered or low-bandwidth links, set it to 1 to serialize downloads and avoid saturating the connection. Every environment is different, so please test and tune to your environment to ensure that your network, firewalls, proxy servers and load balancers are not overloaded.

```
patchwork --download --parallel-downloads 1 --classification CU # Low bandwidth
```

```
patchwork --download --parallel-downloads 6 --classification CU # Fast LAN
```

Note: parallel installation is currently **NOT** possible via the Windows Update Agent API.

### BOUNDING RUN TIME

In scheduled task environments where the task window is fixed, set **--maxruntime** to prevent PatchWork from running past the end of the maintenance window:

```
patchwork --install --classification CU --maxruntime 3600 --autoaccept eula
```

Any updates not reached within the time limit are left for the next run.

### REDUCING SCOPE TO IMPROVE SPEED

Broad filter strings produce larger update sets, which take longer to evaluate and install. Tighten the filters for routine runs, and reserve **--alltypes** for periodic comprehensive scans:

**rem Routine: Critical and Security only**

```
patchwork --install --classification CU --severity CI
```

**rem Monthly audit: everything**

```
patchwork --search --alltypes --classification CUDISRFEVG --xmlout C:\Reports\audit.xml
```

---

## WSUS LOAD

High **--parallel-downloads** values combined with a large **--max-update-count** can generate significant load on a WSUS server and/or network/firewall/proxy infrastructure, especially when run on a large number of devices in parallel. If you are deploying to many machines simultaneously, consider staggering the start times or reducing parallelism. PatchWork's exponential-backoff retry logic (**--retrycount**) will handle transient WSUS server busy conditions gracefully.

---

## 16. TROUBLESHOOTING

---

### --HEALTHCHECK FIRST

Before investigating a failed update run, run **--healthcheck**. It will identify the most common problems — missing WUA service, insufficient disk space, WSUS unreachable, pending reboot blocking installation — in a single pass.

```
patchwork --healthcheck
```

```
patchwork --healthcheck --xmlout C:\Reports\health.xml
```

---

### NO UPDATES FOUND (EXIT CODE 3)

The most common causes:

1. **All matching updates are already installed.** Run with **--history** to confirm.
2. **Filter too narrow.** Try broadening **--classification** or removing **--severity**.
3. **Wrong update source.** If pointing at WSUS, the WSUS server may not have approved updates for this machine. Try **--use-windowsupdate** to compare.
4. **Target group mismatch.** If WSUS is configured with client-side targeting, the machine may be in a group with no approved updates.

---

### WUA COM ERRORS (0X800401F0)

This error means the Windows Update Agent COM class is not registered. The WUA service may be corrupted or disabled. Fix:

```
net stop wuauserv
regsvr32 /s %windir%\system32\wuapi.dll
regsvr32 /s %windir%\system32\wuaueng.dll
net start wuauserv
```

---

### WSUS CONNECTIVITY ISSUES

Verify the WSUS server URL and port. Check that the machine can reach the server:

```
Test-NetConnection -ComputerName wsus.corp.example.com -Port 8530
```

Use **--use-windowsupdate** as a diagnostic bypass. If updates succeed via Windows Update but fail via WSUS, the issue is WSUS-side (approval, targeting, or connectivity).



---

## PROXY ISSUES

If updates fail in environments with a proxy:

- Try **--disable-win-http-proxy** or **--disable-ie-proxy** to check whether the proxy is the cause.
- Try **--auto-detect-proxy** to see if WPAD resolves correctly.
- Use **--debug** to capture proxy negotiation detail in the trace log.

---

## OPERATION TIMEOUT (EXIT CODE 12)

Increase **--maxruntime** or reduce the scope of the run (fewer updates per pass, lower **--parallel-downloads**). On slow or distant WSUS servers, search and download operations take longer; leave extra margin.

---

## READING THE DEBUG TRACE

Enable **--debug**, if requested, to get a full JSON span log at **%TEMP%\patchwork-debug-  
<pid>.log**. This file records:

- The parsed command line after default options are applied
- Every WUA COM call and its HRESULT
- Filter decisions (which updates passed or failed each filter)
- Download progress per update
- Install progress per update
- Error stack traces with context

---

## COMMON WUA HRESULT CODES

| HRESULT    | Meaning  |
|------------|--|
| 0x80240001 | WU_E_NO_SERVICE — WUA not found or disabled              |
| 0x80240003 | WU_E_UNKNOWN_ID — Update ID not recognised               |
| 0x8024000B | WU_E_CALL_CANCELLED — Operation was cancelled            |
| 0x80070005 | Access denied — administrator privileges required        |
| 0x800401F0 | CLASS_E_CLASSNOTAVAILABLE — WUA COM class not registered |

---

## COLLECTING A SUPPORT BUNDLE

To assist with a support request, collect the following:

1. The regular log file (**--logfile** output).
2. The debug trace (**--debug** output from **%TEMP%\patchwork-debug-<pid>.log**).
3. The XML or JSON report from the failing run.
4. The output of **patchwork --healthcheck --xmlout C:\Reports\health.xml**.
5. The output of **patchwork --opt-show** (to confirm the active defaults).
6. The .dmp if automatically generated.

If requested, please send any resulting output to [support@emerita.dev](mailto:support@emerita.dev).

---

## 17. COMPARISON AND MIGRATION

### PATCHWORK VS BUILT-IN TOOLS

| Capability                | wuaclt / UsoClient | PSWindowsUpdate | PatchWork      |
|---------------------------|--------------------|-----------------|----------------|
| Classification filtering  | No                 | Yes             | Yes            |
| Severity filtering        | No                 | Limited         | Yes            |
| Regex pattern filtering   | No                 | No              | Yes            |
| JSON/XML reports          | No                 | Limited         | Yes            |
| Email notifications       | No                 | No              | Yes            |
| Syslog notifications      | No                 | No              | Yes            |
| .NET runtime required     | No                 | Yes             | No             |
| Exit codes for scripting  | Minimal            | Yes             | Yes (extended) |
| WSUS target group control | No                 | Yes             | Yes            |
| KB allow/deny lists       | No                 | Partial         | Yes            |

### MIGRATING FROM PSWINDOWSUPDATE

PSWindowsUpdate uses PowerShell verb-noun syntax. The mapping to PatchWork switches is straightforward:

| PSWindowsUpdate                                     | PatchWork equivalent                |
|---|-------------------------------------|
| <b>Get-WindowsUpdate</b>                            | <b>--search --info</b>              |
| <b>Get-WindowsUpdate - KBArticleID KB5012345</b>    | <b>--search --kb KB5012345</b>      |
| <b>Install-WindowsUpdate - AcceptAll</b>            | <b>--install --autoaccepteula</b>   |
| <b>Install-WindowsUpdate - Category Security</b>    | <b>--install --classification U</b> |
| <b>Remove-WindowsUpdate - KBArticleID KB5012345</b> | <b>--uninstall --kb KB5012345</b>   |
| <b>Get-WUHistory</b>                                | <b>--history</b>                    |

---

## MIGRATING FROM WUINSTALL

WuInstall users will find many switch names familiar ensuring that a change-over to Patchwork is straightforward. PatchWork was designed with similar command-line conventions, although it was **never** designed as a drop-in replacement. There are key differences:

- PatchWork uses **--search**, **--download**, **--install**, **--uninstall** as explicit operation flags rather than positional arguments. “/” is not available to denote switches.
  - Classification codes largely match (**C**, **U**, **D**, etc.) but PatchWork adds **E** (Driver Sets), **V** (Drivers), and **G** (Upgrades).
  - Output formats (XML and JSON) are richer and include per-update descriptions.
  - Default options are stored in the registry via **--opt-save** rather than a configuration file.
-

## 18. APPENDICES

---

### APPENDIX A — FULL EXIT CODE REFERENCE

See [Section 9](#).

---

### APPENDIX B — CLASSIFICATION AND SEVERITY CODE REFERENCE

#### Classification codes:

| Code | Full name          |
|------|--------------------|
| C    | Critical Updates   |
| U    | Security Updates   |
| D    | Definition Updates |
| I    | Updates            |
| R    | Update Rollups     |
| S    | Service Packs      |
| F    | Feature Packs      |
| E    | Driver Sets        |
| V    | Drivers            |
| G    | Upgrades           |

#### Severity codes:

| Code | Full name |
|------|-----------|
| C    | Critical  |
| I    | Important |
| M    | Moderate  |
| L    | Low       |
| U    | Unknown   |

---

APPENDIX C — REGISTRY KEYS

| Key   | Value          | Type          | Written by |
|---|----------------|---------------|------------|
| HKLM\SOFTWARE\Emerita\Patchwork                                   | Installed      | REG_SZ        | --setup    |
| HKLM\SOFTWARE\Emerita\Patchwork                                   | Version        | REG_SZ        | --setup    |
| HKLM\Software\Emerita\Patchwork                                   | DefaultOptions | REG_SZ        | --opt-save |
| HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Environment | Path           | REG_EXPAND_SZ | --setup    |

License keys are also stored in this location, but are not documented here.,

WSUS and proxy configuration keys are modified temporarily during operations and restored on exit.

---

APPENDIX D — SAMPLE XML REPORT

```

<?xml version="1.0" encoding="utf-8"?>
<PatchWorkReport>
  <Summary>
    <TotalUpdates>3</TotalUpdates>
    <TotalSize>314572800</TotalSize>
    <Operation>install</Operation>
    <Status>Success</Status>
  </Summary>
  <Updates>
    <Update>
      <Title>2025-04 Cumulative Update for Windows 10 Version 22H2</Title>
      <KBArticleID>KB5036893</KBArticleID>
      <Classification>Security Updates</Classification>
      <Severity>Critical</Severity>
      <Size>209715200</Size>
      <ReleaseDate>2025-04-08</ReleaseDate>
      <UpdateID>9fb049d9-8ee3-4913-937f-196648006ca5</UpdateID>
    </Update>
  </Updates>
</PatchWorkReport>

```

---

## APPENDIX E — SAMPLE SCHEDULED TASK

The following XML creates a weekly Sunday 02:00 task that runs PatchWork under SYSTEM:

```
<?xml version="1.0" encoding="UTF-16"?>
<Task version="1.2" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
  <Triggers>
    <CalendarTrigger>
      <StartBoundary>2025-01-05T02:00:00</StartBoundary>
      <ScheduleByWeek>
        <WeeksInterval>1</WeeksInterval>
        <DaysOfWeek><Sunday /></DaysOfWeek>
      </ScheduleByWeek>
    </CalendarTrigger>
  </Triggers>
  <Principals>
    <Principal>
      <UserId>S-1-5-18</UserId>
      <RunLevel>HighestAvailable</RunLevel>
    </Principal>
  </Principals>
  <Actions>
    <Exec>
      <Command>C:\Program Files\Emerita\Patchwork\patchwork.exe</Command>
      <Arguments>--install --classification CU --severity CI --autoacceptula --reboot-if-needed --delay 300 --silent --logfile C:\Logs\weekly-patch.log --xmlout C:\Reports\weekly-patch.xml</Arguments>
    </Exec>
  </Actions>
</Task>
```

Import with: `schtasks /create /xml "task.xml" /tn "PatchWork Weekly"`

---

APPENDIX F — SAMPLE POWERSHELL DEPLOYMENT SCRIPT

```
<#
.SYNOPSIS
    Monthly security patching script using PatchWork.
#>

$date      = Get-Date -Format 'yyyyMMdd'
$logFile    = "C:\Logs\patch-$date.log"
$reportFile = "C:\Reports\patch-$date.xml"

# Pre-patch: stop application services
Stop-Service -Name "MyAppService" -ErrorAction SilentlyContinue

# Run PatchWork
patchwork --install `
    --classification CU `
    --severity CI `
    --autoaccepteula `
    --ignore-errors `
    --reboot-if-needed `
    --delay 300 `
    --silent `
    --logfile $logFile `
    --xmlout $reportFile

$result = $LASTEXITCODE

# Post-patch: restart services
Start-Service -Name "MyAppService" -ErrorAction SilentlyContinue

# Report
switch ($result) {
    0 { Write-EventLog -LogName Application -Source PatchWork -EventId 1000 -Message "Patching complete, no reboot." }
    10 { Write-EventLog -LogName Application -Source PatchWork -EventId 1001 -Message "Patching complete, reboot scheduled." }
    3 { Write-EventLog -LogName Application -Source PatchWork -EventId 1002 -Message "No updates found." }
    default {
        Write-EventLog -LogName Application -Source PatchWork -EventId
```



```
1099 -EntryType Warning `
    -Message "Patching finished with unexpected code $result.
See $logFile."
}
}

exit $result
```

---

## APPENDIX G — GLOSSARY

**Classification** — The update category as defined by Microsoft (Critical, Security, Definition, etc.).

**KB** — Knowledge Base article number. Each update is associated with one KB article that describes its content.

**MECM (SCCM)** — Microsoft Endpoint Configuration Manager (formerly SCCM). An enterprise device management platform.

**MU** — Microsoft Update. An update service that extends Windows Update to cover Office and other Microsoft products.

**Severity** — The MSRC (Microsoft Security Response Center) risk rating for a security update.

**WUA** — Windows Update Agent. The operating system component that manages update operations via COM.

**WSUS** — Windows Server Update Services. An enterprise update proxy that caches and controls the distribution of Microsoft updates.

---

## 19. SUPPORT

In the event of any configuration issues or bugs, if you have an active support agreement in place, please contact us via [support@emerita.dev](mailto:support@emerita.dev) so we can investigate further.

For additional support and updates, please visit <https://www.emerita.dev/patchwork.html>

**Patchwork is provided expressly with NO warranty.**

See web address above for terms, conditions and support.

PatchWork is © Emerita Codeworks (UK Ltd) 2026.