Setting Up Your Stream Deck with PowerShell Scripts

A Beginner's Guide for Windows 11 Users

Updated Edition — Including Storm Alerts, Rotator Control, Icons, HamShackFeed & More

---

Overview

This guide walks you through creating PowerShell scripts that launch multiple apps in sequence, send commands to running software, open websites, control hardware, and load standalone web apps — all with a single Stream Deck button press.

What You Will Need

- A Windows 11 PC
- An Elgato Stream Deck (any model) with the Stream Deck software installed
- **Advanced Launcher** plugin by BarRaider — free from the Stream Deck Plugin Store
- The apps you want to launch already installed on your PC
- Basic comfort with copy/paste and editing a text file
- Python 3.x (optional — only needed for the local web server feature in Step 7)

Why Advanced Launcher? The built-in System > Open action may not include an Arguments field on all Stream Deck versions. Advanced Launcher reliably provides Arguments, working directory, and Run as Administrator.

Step 1 — Find the Path to Each App

Right-click the app shortcut, choose Properties, click Shortcut tab, copy the Target field:

```
C:\Program Files\SomeApp\SomeApp.exe
```

Tip: *No shortcut? Search Start Menu, right-click the app, Open file location, then right-click the .exe.*

Step 2 — Create a Folder for Your Scripts

```
C:\My Scripts\
```

Step 3 — Create the PowerShell Script

Open Notepad and paste this template:

```
$app1 = "C:\Program Files\YourApp1\YourApp1.exe"
$app2 = "C:\Program Files\YourApp2\YourApp2.exe"
$shortDelay = 3

function Launch($path, $name) {
```

```
    if (Test-Path $path) {
        Write-Host "Launching $name..." -ForegroundColor Green
        Start-Process $path
    } else {
        Write-Host "WARNING: $name not found" -ForegroundColor Yellow
    }
}

Launch $app1 "App One"
Start-Sleep -Seconds $shortDelay
Launch $app2 "App Two"
Write-Host "All done!" -ForegroundColor Green
Start-Sleep -Seconds 3
```

Adding a Reminder Popup

```
Add-Type -AssemblyName PresentationFramework
[System.Windows.MessageBox]::Show(
    "Check settings before continuing.`n`nClick OK when ready.",
    "Launcher - Action Required",
    [System.Windows.MessageBoxButton]::OK,
    [System.Windows.MessageBoxImage]::Warning
) | Out-Null
```

Launching as Administrator

```
Start-Process $path -Verb RunAs
```

With working directory:

```
Start-Process $path -WorkingDirectory "C:\Program Files\YourApp\" -Verb RunAs
```

Multiple Modes in One Script

```
param ( [string]$mode = "default" )
switch ($mode.ToLower()) {
    "mode1" { Launch $app1 "App One"; Start-Sleep -Seconds $shortDelay; Launch $app2 "App Two" }
    "mode2" { Launch $app3 "App Three" }
}
```

Step 4 — Test the Script

```
powershell.exe -ExecutionPolicy Bypass -File "C:\My Scripts\MyLauncher.ps1"
```

If you see a digital signature error:

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser
```

> ***Always test in a visible PowerShell window first.*** *Errors that are silent from Stream Deck will show in PowerShell.*

Step 5 — Set Up the Stream Deck Button

Use the Advanced Launcher plugin:

- Application: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

- Arguments: -ExecutionPolicy Bypass -File "C:\My Scripts\MyLauncher.ps1"
- For multiple modes, append: -mode mode1
- Check Run as Administrator if the script requires it
- Optionally assign a custom icon (see Step 11)

Step 5b — Hotkey Switch for Keyboard Shortcuts

Some Windows features are best triggered with a keyboard shortcut. The Hotkey Switch action sends a key combination as if pressed on the keyboard.

Example — Snipping Tool (Win+Shift+S):

- Drag Hotkey Switch onto a button
- Click the key combination field
- Press Windows + Shift + S
- The shortcut is recorded

Other useful shortcuts: Win+V (Clipboard), Win+D (Desktop), Win+L (Lock)

Step 6 — Opening Chrome with URLs

Multiple tabs via script:

```
$chrome = "C:\Program Files\Google\Chrome\Application\chrome.exe"
$urls = @("https://www.example1.com","https://www.example2.com")
$args = '--profile-directory="Profile 1" ' + ($urls -join " ")
Start-Process $chrome -ArgumentList $args
```

Single URL — no script needed in Advanced Launcher:

- Application: C:\Program Files\Google\Chrome\Application\chrome.exe
- Arguments: --profile-directory="Profile 1" https://www.yoursite.com

Find your Chrome profile name at chrome://version under Profile Path.

Step 7 — Running a Local Web Server for Dashboards

Serve local HTML dashboards via Python to allow live data fetching without CORS errors.

```
$dashboardFolder = "C:\My Scripts"
$port = 8073
$url  = "http://localhost:$port/MyDashboard.html"

$python = $null
foreach ($cmd in @("python","python3","py")) {
    try { if ((& $cmd --version 2>&1) -match "Python") { $python=$cmd; break } } catch {}
}

$inUse = netstat -ano | Select-String ":$port " | Select-String "LISTENING"
if (-not $inUse) {
    Start-Process $python -ArgumentList "-m http.server $port" `
        -WorkingDirectory $dashboardFolder -WindowStyle Hidden
    Start-Sleep -Seconds 2
}
Start-Process "chrome.exe" -ArgumentList '--profile-directory="Profile 1"',$url
```

Port 8073 is easy to remember for ham radio — 80.73 meters.

Step 8 — Opening Windows System Tools

Open .msc files directly in Advanced Launcher:

```
Application: C:\Windows\System32\devmgmt.msc
Arguments:   (leave empty)
Run as Administrator: Yes
```

- **Device Manager:** C:\Windows\System32\devmgmt.msc (Run as Admin)
- **Services:** C:\Windows\System32\services.msc (Run as Admin)
- **Event Viewer:** C:\Windows\System32\eventvwr.msc
- **Task Manager:** C:\Windows\System32\taskmgr.exe
- **Control Panel:** C:\Windows\System32\control.exe

For .cpl applets like Sound, use PowerShell:

```
-ExecutionPolicy Bypass -Command "Start-Process mmsys.cpl"
```

Other .cpl files: appwiz.cpl (Programs), desk.cpl (Display), timedate.cpl (Date/Time)

Step 9 — Sending Commands to Software via UDP

Send control commands to running software that supports UDP remote control.

```
param ( [int]$az = -1 )
$udpHost = "127.0.0.1"; $udpPort = 12000
if ($az -lt 0 -or $az -gt 360) { Write-Host "Usage: .\Script.ps1 -az 270"; exit }
$command   = "<PST><AZIMUTH>$az</AZIMUTH></PST>"
$udpClient = New-Object System.Net.Sockets.UdpClient
$udpClient.Connect($udpHost, $udpPort)
```

```
$bytes = [System.Text.Encoding]::ASCII.GetBytes($command)
$udpClient.Send($bytes, $bytes.Length) | Out-Null
$udpClient.Close()
Write-Host "Sent: $az degrees" -ForegroundColor Green

-ExecutionPolicy Bypass -File "C:\My Scripts\Rotator.ps1" -az 270
```

Verify a port is free: netstat -ano | findstr ":12000"

Step 10 — Organizing with Pages and Folders

Pages: Click + at the bottom of the Stream Deck software to add pages. Swipe between them on the device.

Folders: Drag Create Folder onto a button to make a sub-page with an automatic back button. Great for rotator directions, programming tools, or system utilities.

Keep most-used buttons on page 1. Use folders for related groups.

Step 11 — Custom Icons

Apply a custom icon: click the button, click the icon thumbnail, browse to your PNG. Specs: 144x144 RGBA PNG with ~16px rounded corners.

Generate icons with Python (pip install Pillow):

```
from PIL import Image, ImageDraw, ImageFont
SIZE = 144
img = Image.new('RGBA', (SIZE,SIZE), (0,0,0,0))
d   = ImageDraw.Draw(img)
d.rounded_rectangle([0,0,143,143], radius=16, fill=(10,22,40,255))
d.rounded_rectangle([2,2,141,141], radius=15, outline=(0,180,255,120), width=2)
font = ImageFont.truetype("C:/Windows/Fonts/arialbd.ttf", 28)
d.text((72,80), "FT8", font=font, fill=(0,220,255,255), anchor='mm')
img.save("FT8.png")
```

Step 12 — Opening a Standalone HTML App (HamShackFeed)

Not every Stream Deck button needs a script. Some tools are built as standalone HTML files that open directly in Chrome with no server or installation required. HamShackFeed is one example — a ham radio content aggregator that pulls together blogs, podcasts, and YouTube channels into one searchable dashboard.

What HamShackFeed Does

- 17 pre-configured ham radio sources — blogs, podcasts, and YouTube channels
- Color-coded cards sorted newest first (cyan=blog, gold=podcast, red=video)
- Filter by content type — All / Blog / Podcast / Video
- Read tracking with unread counts that persist between browser sessions
- Add sources by pasting any blog, podcast, or YouTube URL
- Remove sources with one click from the sidebar
- Real-time search plus DuckDuckGo web search for longer queries
- No server, no account, no installation — just open in Chrome

Why No Server Is Needed

HamShackFeed uses the free rss2json.com API to fetch RSS feeds server-side, working around browser CORS restrictions without needing a local Python server. This is different from the propagation dashboard, which requires Python because it fetches data in more complex ways.

Stream Deck Button Setup

Save HamShackFeed.html to a subfolder:

```
C:\My Scripts\HamShackFeed\HamShackFeed.html
```

Advanced Launcher settings — no script needed:

- Application: C:\Program Files\Google\Chrome\Application\chrome.exe
- Arguments: --profile-directory="Profile 1" "C:\My Scripts\HamShackFeed\HamShackFeed.html"
- Run as Administrator: No

Adding Sources

Click + Add Source and paste any URL. HamShackFeed auto-detects the type:

- Blog — paste the website URL, /feed is tried automatically
- Podcast (Podbean/Spreaker) — feed URL constructed automatically
- YouTube — channel ID extracted from the URL for the RSS feed

Added sources and read state are saved in browser local storage — they survive page refreshes and browser restarts.

Sharing HamShackFeed

Because HamShackFeed is a single self-contained file, sharing it is as simple as sharing any other file. Other users get the full default source list and can customize it independently — their changes are stored locally and don't affect anyone else's copy.

Download: github.com/N4MI73/streamdeck-hamradio

Tips and Tricks

- **Adjusting delays:** If an app is not loaded before the next launches, increase the delay. Start at 3-5 seconds.
- **Working directory matters:** Some apps fail if started from the wrong folder. Add -WorkingDirectory to Start-Process if an app shows runtime errors when launched from a script.
- **Always back up before editing:** Before changing a working file, copy it first. Especially important for HTML files which can be silently overwritten as empty files if a script error occurs.
- **Keep GitHub in sync:** Push changes after significant updates. When a file is accidentally overwritten, the last good version is always retrievable from the repository.
- **Testing port availability:** netstat -ano | findstr ":12000" — no output means the port is free.
- **PowerShell 7:** Install alongside 5.1 with no conflicts: winget install Microsoft.PowerShell
- **Getting help:** Claude at claude.ai can write or troubleshoot PowerShell scripts. Describe what you want, share file paths and software names, and Claude will build the script.

Guide prepared with assistance from Claude (claude.ai)