

Keyboard manager

- **What is it:** Users can remap their keys
- **Authors:** Sakariya Ahmed
- **Spec Status:** Review



1. Overview

1.1 Key definitions

Here are a few definitions of words found throughout this document to ensure clarity:

- **Key Remapping:** Refers to the act of changing the output of a keystroke on your keyboard. An example is remapping the "Tab" key to "Delete".
- **Shortcuts or Key shortcuts:** A combination of 2 or more keystrokes that results in a user-defined action. An example is `Ctrl+Z` un-doing an action in Microsoft Word
- **KSM:** Acronym for Keyboard Shortcut Manager
- **Keystroke:** A single press of a key
- **Exotic key:** These are keys that are unique and don't exist on many keyboards. Example: Calculator or search button.

1.2 Press Release

PowerToys has just released a new utility called "Keyboard Shortcut Manager"! This keyboard manager enables customers to customize their computer on a keyboard level. With it, all the keys on your keyboard are dynamic and can be remapped to provide different outputs.

Ever switched computers and the shortcut to search your computer was different? Or change the keys you press an app you use often? This utility is perfect if you're always looking to be more efficient on your computer or are a long-time user of a different OS and want to bring your keypresses and shortcuts with you!

1.3 Elevator Pitch

The keyboard is still the primary method of interacting with your device(s). Historically, Windows has not easily let you configure your keyboard. The new PowerToys Keyboard Manager allows you to customize the keyboard to be more productive by remapping keys and creating your own keyboard shortcuts.

1.4 Tweet


"Ever wish you could change some shortcuts and edit keyboard buttons? Or switch computers and your keyboard was slightly different? PowerToys is here to solve all your keyboard woes with a Keyboard Shortcut Manager! Customize your experience by remapping keys or creating new shortcuts!"

1.5 Narrative / Scenario

Switching from Mac/Linux: A user who has recently made the switch to Windows and is having trouble adjusting to the keyboard differences. Simple tasks like locking the computer, copy / paste, and switching browsers now require thinking and the lack of muscle memory makes the user feels clumsy at times. For example, copying an object on MacOS, `cmd + C`, requires your thumb whereas on Windows, `Ctrl + C`, it's your pinky. Even within identical

applications there are shortcut disparities. Going to the address bar in Google Chrome on a Mac is `cmd + L`, on a PC, that would map to `Alt + D`.

Outcome for Switching: A successful KSM manager would increase comfortability and users wouldn't feel like they lost functionality or convenience with switching to Windows 10. In effect, it would help decrease churn and avoid a negative experience for a new user, which is crucial moment for public sentiment on Windows.

Customize my keyboard – Users want customizability to make them faster: Developers, or power users, who use their keyboard a lot are constantly looking to save themselves time and make their processes more convenient. This community has largely relied on 3rd party tools or has left their keys alone but voiced frustration with the lack of a native tool to remap keys. An example of a power shortcut that made its way into Windows is  Windows Key + L.

You used to have to hit `Ctrl + Alt + Del` then click "Lock".

Outcome for Customizing: A successful KSM manager would 'unlock' functionality and convenience for many power users, increasing their everyday efficiency. Keyboard changes are highly visible and successfully extending control would greatly boost satisfaction to continue to cement Windows as the *ideal* platform for a developer.

1.6 Customers

We have two primary customers:

- **Developers/PowerUsers** of Windows10 that want a built-in and lightweight solution to remap their own hotkeys which will increase their efficiency in completing tasks.
- **Former Linux and MacOS** users who now use Windows and would like to utilize their former keyboard shortcuts to get rid of the learning period and ease the transition to using a new device.

1.7 Problem Statement and Supporting Customer Insights

Windows 10 is available on a vast array of devices using both physical and touch screen keyboards for user interaction. Lack of keyboard customization is leading to user frustration when switching from another OS and satisfaction around being less productive.

Comfortability/Ease of use, especially for new users, can be crucial as their first few interactions can cause frustration and result in a higher churn towards competitors. For developers and some seasoned users, where using their keyboard is a large part their job, ability to remap keystrokes and engage executables can lead to massive gains in time. In fact, this was the [second most popular topic](#) measured through thumbs up and the most commented issue in the PowerToys Github.

1.8 Solutions or Expectations






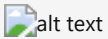





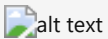



For casual users, they are forced to use Windows shortcuts, if anything, and forgo additional productivity due to a lack of a keyboard mapping solution built-in to Windows. Whereas for users where this has become an ongoing concern, the vast majority who switch to the Powertoy will come with certain expectations as they have come to rely on a multitude of 3rd party solutions. 28% of survey respondents used 3rd party tools, and 60% of those who did used either AutoHotKey or SharpKeys.

Various 3rd party solutions:


The following are the most popular 3rd party apps across different operating systems:


- AutoHotkey
- SharpKeys

- Karabiner (MacOS)

	AutoHotKey	SharpKeys	Karabiner
Does the application have a UI?			
OS-Level Shortcuts			
App-Level Shortcuts			
Multi-keyboard support			
Profiles for different uses			

Sharpkeys is an open source application created for Windows that allows the user to remap keys through their computer's registry. It was created to make the process of accessing the registry and reprogramming keys more convenient for advanced users and accessible for casual users. The Sharpkeys GUI (*Figure 1*) allows users to keep track of remapped keys and import/export key configurations. When users try to remap a key, they can do it through pressing the key on their keyboard or selecting the key from a drop-down menu. Both are exhaustive for North American users and the SharpKeys community is currently working to expanding their support for even more international keyboards. Sharpkeys requires a restart but doesn't require elevated permissions or to run in the background. It does not support the remapping of key combinations or shortcuts. Additionally, it is unable to remap keys that cannot be remapped from Windows registry access: Fn, Windows, Calculator, etc.

 alt text Figure 1. SharpKeys: Landing Screen


 alt text Figure 2. Sharpkeys: Key Remapping Input


AutoHotKey (AHK) is a scripting language that runs on Windows and while it focuses on and was first created to remap keys and create shortcuts, it is capable of a lot more. Using AHK requires spending time reading their extensive documentation (*Figure 3*) to become adept at writing scripts. There is no GUI or interface that comes with AHK and it is all interpreted through a text editor. With AHK users can create both OS-level and app-specific shortcuts, however there is no simple way to share/save key configurations among multiple users. AHK has a very strong community that has been able to create tools that fill in many of the language's functionality gaps.

 alt text Figure 3. AutoHotKey Documentation

Karabiner is a MacOS solution for users to remap their keys. Karabiner's interface (*Figure 4*) is intuitive and accessible to both beginner and advanced users. It is capable of what it calls "simple modifications, which are just remapping a single key to another key. Karabiner has support for multiple keyboards and the user can target a specific device or all devices for any remapping. Complex modifications (*Figure 5*) are defined by Karabiner as creating/editing shortcuts that consist of multiple key presses or mapping a shortcut to a single key. These complex modifications can be imported from Karabiner's website where the community has created a vast array of key configurations. If you cannot find a suitable configuration, a user can create their own by using the terminal to open Karabiner's JSON file.

After learning the syntax from the reference manual, this JSON file can be edited to automatically add new user-defined rules.

 alt text Figure 4. Karabiner: Landing Screen

 alt text Figure 6. Karabiner: Complex Modifications

1.9 Survey Results

A brief survey was sent to the Windows Developer community during the planning stage of this utility. The consumer research was then used to inform this functional spec and test the preliminary hypothesis. Here's a summary of the survey results

- *Question:* Do you currently use any third-party tools to remap keys and/or create shortcuts?
 - Nearly 30% of survey respondents used 3rd party tools. This indicates that there is not only a real demand for the utility and many users currently remap their keys.
- *Question:* Which third party tools do you use to remap keys and create shortcuts?
 - The most popular 3rd party tool was AutoHotKey, followed by SharpKeys. Karabiner was not mentioned as this was sent to Windows developers but it was still highlighted in the competitive research due to its popularity among Mac users.
- *Question:* If you could improve the way that you manage your remap keys and create shortcuts using the third-party tools, what would you change?
 - Over 50% of survey respondents asked for either native integration into Windows and / or a pleasant UI experience instead of scripting. This confirmed the decision to ensure users can engage with all of the utility's functionality through a UI.
- *Question:* What features would you most like in a keyboard shortcut manager besides remapping keys?
 - ~70% of users requested shortcuts, either app-level or OS level, with OS shortcuts being the most demanded. This helped us prioritize the different features we planned in the functional spec.
- *Question:* If remapping your keyboard required a restart, would you still use this PowerToy?
 - Before design, we were intimately aware that this could be a pain point for many users. Over 60% of respondents answered 'Yes', providing confidence that a restart, if necessary, is possible.

1.10 Goals / Non-Goals

Goals

- The KSM can be used to configure keypresses on Windows 10 with any built-in / external keyboard
- The KSM provides an intuitive and lightweight UI experience (53% of survey respondents who use 3rd party tools wanted integration into Windows and/or a UI)
- The KSM will be able to remap keys
- The KSM will be able to edit OS-level shortcuts
- The KSM will work with all keyboards, regardless of locale (Example: Japanese keyboard)

Stretch Goals




- The KSM will be able to edit app-level shortcuts

Non-Goals

- Require a reboot only when absolutely necessary (42% of survey respondents would not use this PowerToy if reboot was required)

2. Requirements

2.1 Functional Requirements Overview

No.	Requirement	Pri
1	Remap any button on keyboard to any other button on keyboard	P0
2	OS-Level key shortcuts. Example: Making  + G lock the computer.	P1
3	App level keyboard shortcuts. Example: Going to the address bar in Chrome on a MacOS is <code>cmd+L</code> , on a PC, that would map to <code>Alt + D</code> .	P2
4	Settings page where user can change key remapping/keystrokes and other supported changes.	P1
5	The Powertoys Shortcut Guide needs to be aware of keyboard remaps. Example: If  + G locks the computer the Shortcut Guide needs to list this as the method to lock a computer, not  + L.	P1
6	A Mac ready profile that is prepopulated with commonly used remappings and adjustments.	P1
6	A Linux ready profile that is prepopulated with commonly used remappings and adjustments.	P1
7	The KSM needs to meet Microsoft accessibility requirements.	P1
8	The KSM needs to meet Microsoft localization requirements.	P1
9	Multiple keyboard support. Example: Surface Laptop and Surface keyboard have different layout and a remap maybe needed*	P3


***MULTIPLE KEYBOARD SUPPORT WAS MOVED DOWN IN PRIORITY AS THERE ARE CURRENTLY NO PUBLIC APIS THAT CAN ACCOMPLISH THIS. AS A RESULT WE'RE WORKING ON HAVING THOSE APIS AVAILABLE DOWN THE ROAD, BUT IT WILL NOT BE PART OF OUR V1 PRODUCT.**

2.2 Functional Requirements for remapping of keys

No.	Requirement	Pri
1	Remapping of keys is done through a user interface. User is able to select the key to remap and the corresponding key to remap it to.	P1
2	User can edit both global and key-board specific shortcuts. Example: See Karabiner	P2
3	User can remap the function key.	P3
4	User can remap keys through a JSON file	P3

2.3 Functional Requirements for OS-Level shortcuts



No.	Requirement	Pri
-----	-------------	-----

1	Can disable any existing OS-level shortcuts. Example:  Windows Key + C	P0
2	OS-level shortcuts have priority over apps that come with shortcuts. Example: Alt + Enter will trigger OS action, not excel-specific action when in Excel	P1
3	Winkey shortcuts have priority over app-specific shortcuts	P2

2.4 Functional Requirements for App-Level shortcuts (stretch goal)

No.	Requirement	Pri
1	A list of available apps on computer with <i>friendly names</i>	P2
2	Assign app-level shortcuts through settings UI	P2
3	Able to set global shortcut exceptions to certain apps	P3

2.5 Functional Requirements for Settings


No.	Requirement	Pri
1	Recognize key presses.	P0
2	Clearly show what keys cannot be remapped	P1
3	Option to reset to default	P1
4	Save, load, and export KSM settings so they can be loaded onto a different computer	P1
5	Warn users if any Winkey shortcuts have been orphaned. Example: Mapping  Windows Key + D to  Windows Key + L in one-direction, ergo no longer providing the user a method to lock their PC.	P1
6	Display all remapping and shortcut changes that have been made	P1
7	Settings are capable of dark mode	P2
8	Warn users when multiple keys have the same function. Example: Left Alt and Right Alt	P2
9	I am presented with a list of all the current OS-level key mapping and shortcuts in use	P3
10	Users can quickly swap profiles for different workloads. Example: See #2	P3

2.6 Functional Requirements Open Questions / Concerns

2.6.1 Circular reference

- When two or more keys are remapped to each other the logic needs to ensure there is never a loop and clear indication regarding priority order of operations.

2.6.2 WinKey Priority and Access

- Should certain shortcuts be restricted to the user? Example:  Windows Key + D

2.6.3 Multiple Keyboards – Shortcut support

- Currently, there exists no plan to incorporate the creation of shortcuts for specific keyboards. Is this an expected use case?

2.6.4 SharpKey / AHK Partnership

- SharpKey has weak copy-left license (Ms-PL Microsoft Public License) and we need to evaluate whether any of its codebase is useful. AutoHotKey's scripting model could help us but we need to validate if its possible / if it's a good idea. 52% of survey respondents who used 3rd party tools used AHK and 68% used either AHK or Sharpkeys.

2.6.5 Elevated Permissions

- Is consistently running elevated (aka run as admin) required for the Keyboard Shortcut Manager?

2.6.6 F-Key / Fn key ignore

- Can we have a setting that makes `Fn` keys (i.e `F1-F12`) always act like function keys?

3. Measure Requirements

No.	Requirement	Implication	Pri
1	# of keys being remapped per user and which keys	Insight into use and whether certain keys are persistent across the community. Allows us to identify key recommendations and keys that are causing consistent issues.\	P1
2	# of OS-level shortcuts created vs # of App-level shortcuts	Understand how users are using shortcuts and about which shortcuts are critical.	P1
3	# of times the same user loads different key configurations.	Answers where multi-profile use, as a new feature, should be prioritized	P1
4	# of times built-in shortcuts are disabled	Allow us to recommend defaults to Windows developers.	P1
5	# of times built-in shortcuts are remapped	Allow us to recommend defaults to Windows developers.	P1
6	Specificity and frequency of <i>exotic</i> keys being remapped.	Whether the KSM is missing keys and if some keys are not useful (i.e being mapped over)	P1
7	# of visits to the KSM editor/settings in a given time period	Continuous visits in a short time could imply lack of clarity with UX or bug in app	P1
8	Frequency of keys being mapped to each other (swapped). Example: <code>Key A -> Key B</code> , <code>Key B -> Key C</code> , <code>Key C -> Key A</code>	Whether auto/hot swapping, as a new feature, needs to be prioritized	P1
9	Amount of people that have remapped turned on / off	Information on whether KSM is being utilized by PowerToys users.	P1
10	Maximum number of keyboards someone has attached to their keyboard in a 28-day period	Validates whether to prioritize additional keyboard support.	P1

4. UI Mockup

See below for gifs/screenshots of the UI mockups:

*Please understand that these are preliminary and requirements that not fully defined may be in the design and others may be missing**

Below are *perliminary* screenshots and gifs of the app. This is being changed frequently and is simply a show of the basic functionality we hope for our v1.



4.1 Main UI Settings



4.2 Main UI Settings Cont'd



4.3 Remapping Keys Settings



4.4 Edit Shortcuts Settings



4.5 Detecting Shortcut Key Presses

