How to create a time tracking application on Windows with AutoHotKey

This application can keep track of all the windows you have used during the day. You only need AutoHotKey, a basic word processor like Notepad, and about half an hour.

You start work every day early so you can handle everything for the day. Then, you check the clock, it's 3 am but the work is still not done. So where has the time gone?

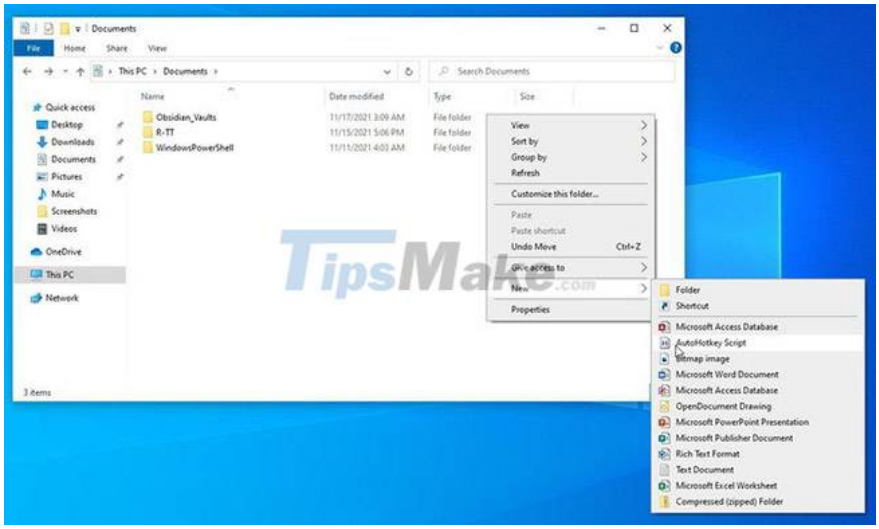
You can use a dedicated time tracking app, but these apps can be complicated and frustrating. Instead, why not create a simple little app of your own? This application can keep track of all the windows you have used during the day. You only need AutoHotKey, a basic word processor like Notepad, and about half an hour. Let's get started!

Create your own window logging script with AutoHotKey

Before you start, you should install AHK on your computer as it will act as the "parser" for the script. It's the tool that will allow your script to "run".

> Note: You can also compile the script once it's done to turn it into a real executable. However, that is beyond the scope of this article.

Download AutoHotKey from the official website here and install it.



Select New > AutoHotKey Script

Fire up your favorite file manager and go to the folder where you want to store the script. Then, right-click an empty spot and select New > AutoHotKey Script.

Once done, it's time to write the actual script.

1. Define the required variables

Open the script in your favorite editor. You can use something as simple as Notepad that comes with Windows, but the article will use Notepad++ for this tutorial. Notepad++ is free and better adapted for this purpose, so you should give it a try.

Note that you should not use any applications such as Word or Google Docs, this may affect the formatting of the script. Use a text or code editor.

The script will contain some basic recommended information about compatibility and performance. Leave them as is and start the script below.

Starts with:

```
AppLoggingRate = 10 ; Time interval (in seconds) between active window title cap
```

Start by assigning the value "10" to AppLoggingRate, which will be used to calculate the time between window logging.

When used with the AHK's Sleep function, 1000 is close to one second. So by multiplying it by the AppLogingRate, you will make the SleepTime variable "equal to 10 seconds".

LogPath is the path used to store the logs. The example is using the %A_ScriptDir% value, which is translated to "the directory from where you run the script". You can use the full path to another directory if you want.

Finally, set LastActiveWindow to empty and use the following to check if the active window has changed.

2. Monitor active windows

Since we want to continuously monitor which window is active and, if it changes, record the title and time, we will have to use a 'loop'.

As the name suggests, a loop runs continuously, repeating the same function(s). Thanks to AHK's simple syntax, the following "code" is relatively easy to understand:

```
Loop { Sleep %SleepTime% Msgbox, It Works! }
```

Define a loop by simply typing the word "loop" and then marking its beginning with "{" and ending with "}". Anything on the lines between "{" and "}" will run forever until you exit the script.

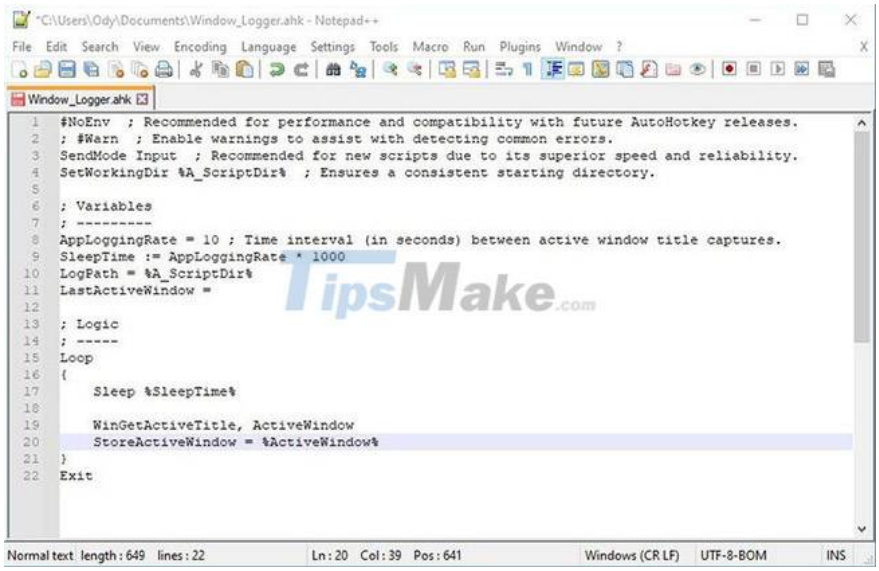
Start the loop by waiting (Sleep) for an amount of time equal to the SleepTime variable. This variable makes time control simpler. Instead of manually editing the script, you can "tell" it, through this variable, how many seconds each loop should last.

Finally, use the Message Box to test the script. Try saving and running it (double click on the script file). You should see a message box that says 'It Works!' (script is working) after 10 seconds.

Right click on the icon of AHK in the Windows tray and exit the script when you have enough message boxes. Then go back to the editor and replace the MsgBox line with:

```
WinGetActiveTitle, ActiveWindow
```

This is the command to get the title of the active window. Omit the extra "StoreActiveWindow" line that the example used while writing the test script.



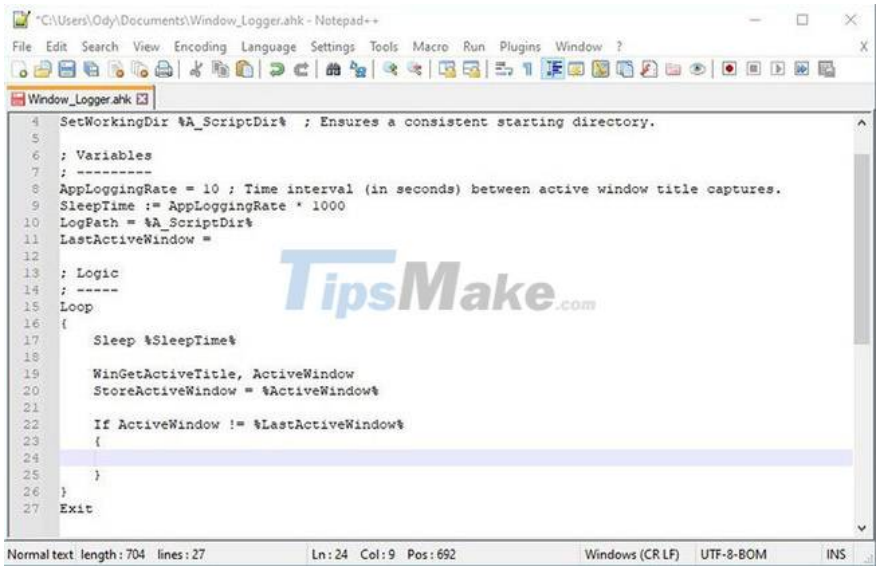
Get the active window's title and store it in a variable

3. Get current time and name

Now to the core of the script logic. Compare the active window's name with the previous one, and if they're different, "do something". Everything is as simple as this:

```
If ActiveWindow != %LastActiveWindow% { }
```

Given the above, check if the current ActiveWindow is different (!=) from the value stored in the LastActiveWindow variable (which was initially set to empty). If so, AHK will execute the code between { and }, which is currently empty.

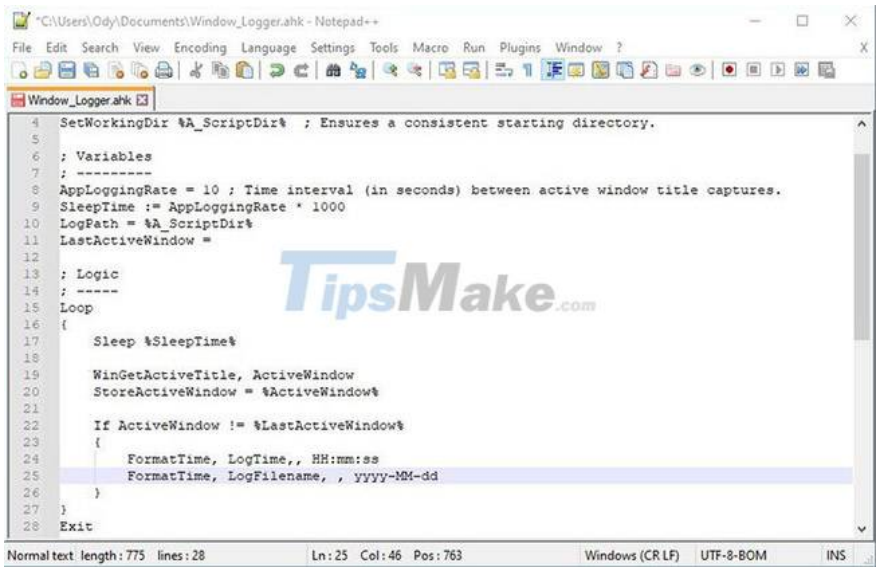


Set the function to compare the title of the active window and the previous window

Both date and time are needed to measure the uptime of a window. Different logs for each day, using the date in their name will be kept. Not only every change but when it happens will also be recorded. To do that, assign different time formats to the LogTime and LogFilename variables, with:

```
FormatTime, LogTime,, HH:mm:ss FormatTime, LogFilename,, yyyy-MMM-dd
```

Add those lines between the curly braces in "If ActiveWindow.", so that AHK runs them when it detects a window change.



Get the current time and assign it in two variables of different format

4. Data Format

We have now captured the time in two differently formatted variables, as well as the title of the active window. However, there is a small problem: The title of the window can also contain unwanted characters. All non-

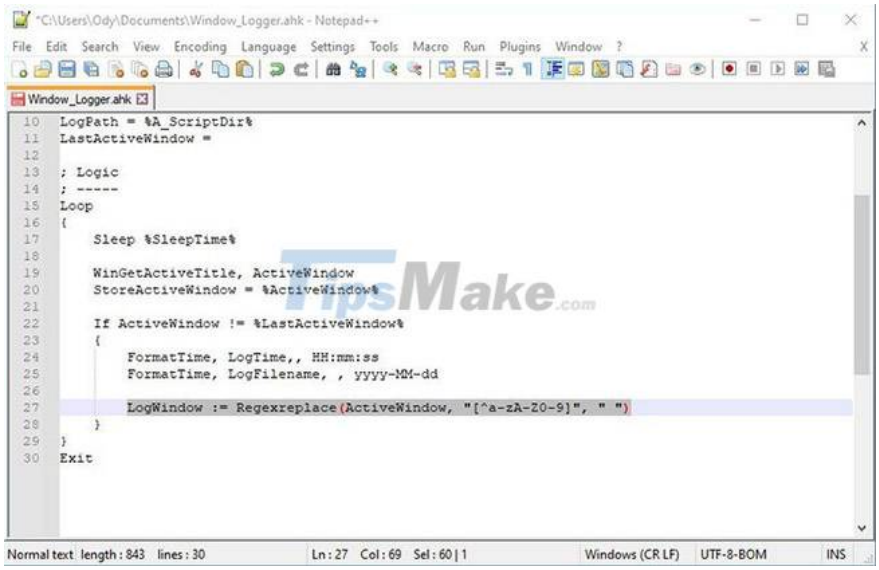
alphanumeric characters can be removed using AHK's support for RegEx, with:

```
LogWindow := Regexreplace(ActiveWindow, "[^a-zA-Z0-9]", " ")
```

With this, "ask" AHK to remove all characters from the ActiveWindow variable that don't match what's in the brackets:

1. Lowercase
2. Uppercase letter
3. Numbers

Then assign the result to the LogWindow variable.

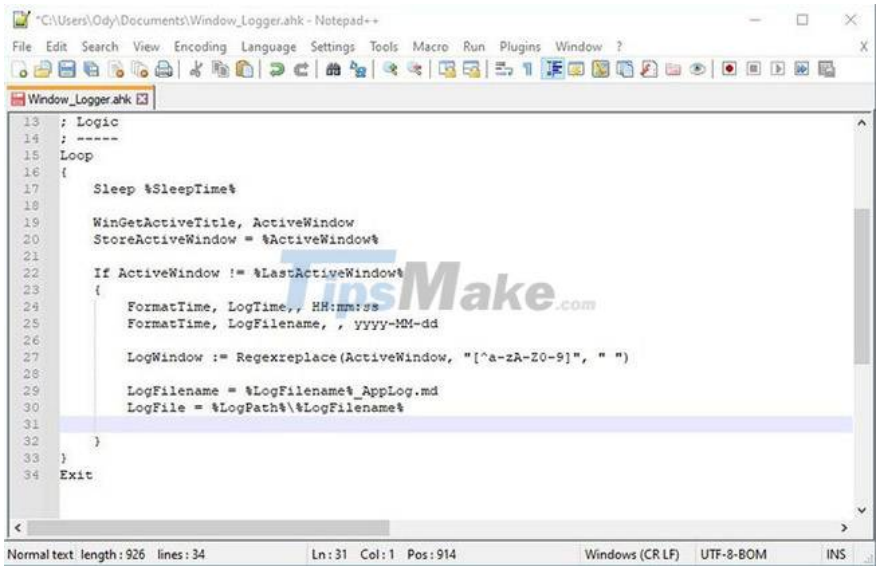


Clean up the active window's title with RegEx

With all variables set and all valuable data collected, you are now ready to format the log file and its contents.

```
LogFilename = %LogFilename%_AppLog.md LogFile = %LogPath%%LogFilename%
```

Previously, we assigned the current date to the LogFilename variable. So for the first line, add "_AppLog.md" to the date to use it as the filename.



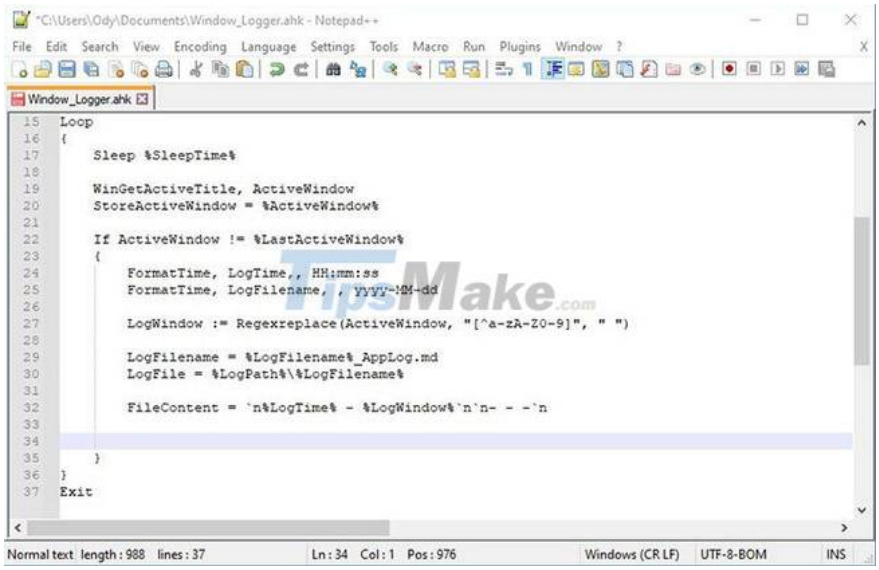
Set log file name

In the second line, incorporate the LogPath variable, specified at the beginning as the destination for the log file associated with the filename. Their combination is the full pathname of the log file, which is assigned to the LogFile variable.

Assign the equivalent of "empty line, Time - Window's Name, two more empty lines, a divider, and another empty line, for good measure" to the FileContent variable.

```
FileContent = `n%LogTime% - %LogWindow%`n`n- - -`n
```

1. The "n" tells AHK to enter a new line (equivalent to pressing Enter once).
2. The three dashes will appear as a separator when displayed in a down-compatible viewer.
3. "%LogTime%" and "%LogWindow%" are variables that have stored the name of the active window and the time it was detected.



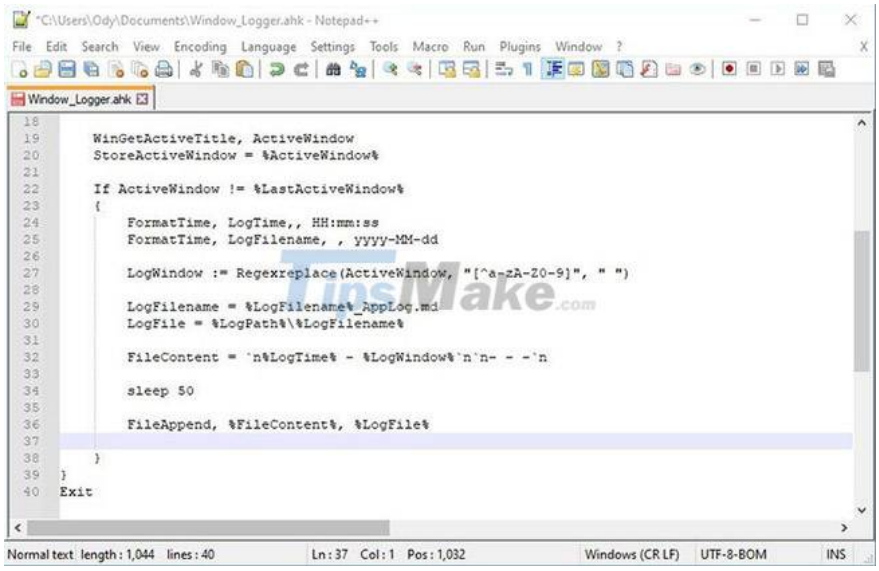
Determine the contents of the log file

5. Update files

You have specified what we want to write to the file, as well as its path and filename. All that remains is the actual text, as simple as this:

```
FileAppend, %FileContent%, %LogFile%
```

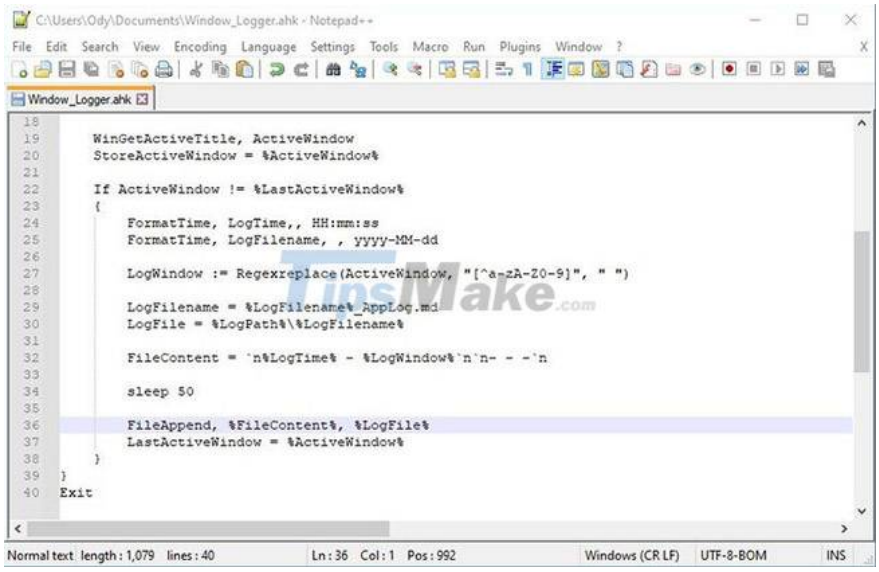
Append everything in the variable "FileContent" to the file "LogFile".



Use AHK's Append function to update the log file or create one from scratch

The "append" function will add "FileContent" to the file if it exists, but will also create it from scratch if the file doesn't exist.

There is one final tweak: replace the content of the LastActiveWindow variable with the currently active window.

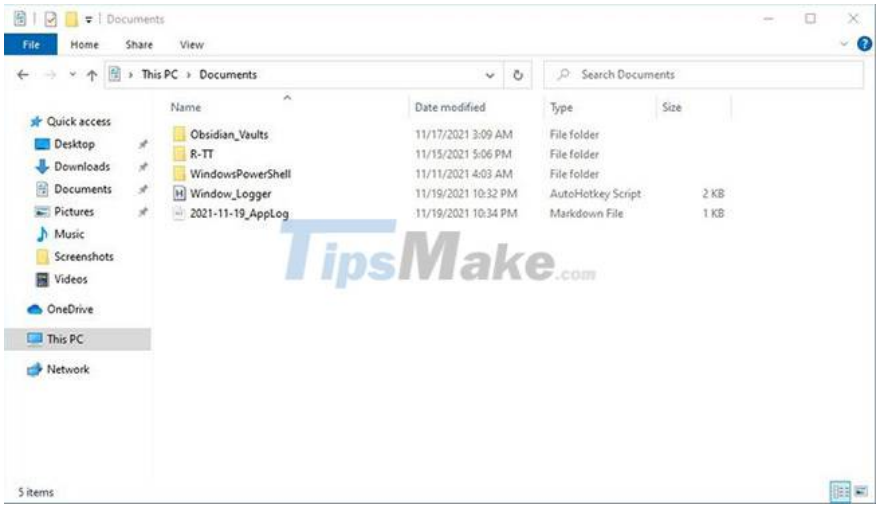


Insert title of current active window into LastActiveWindow variable for future checking

To do this, the script should be able to detect the next window change.

```
LastActiveWindow = %ActiveWindow%
```

And with that last addition, the window logger is ready! Save the script and run it. Then check the markdown file, it will appear in the script files folder after 10 seconds.



The windows log file is created in the same directory as the script

Own your own time

You can open the log file with any text editor. However, it will look better if you open in a markdown compatible editor. In the screenshot you can see the log in the popular Typora editor.

The markdown file generated by the script loaded in Typora

It's an easy way to check which apps you've been using the most time in, and it just needs a tool like Notepad to use.

If you want something "more special", you can always "stylize" the logger output to generate a CSV file instead. It's as easy as adjusting the FileContent variable and the extension of the created file. You can then import such files into applications like Excel, Google Calc or even third-party time trackers.

Complete script

```
#NoEnv ; Recommended for performance and compatibility with future AutoHotkey re
```