UserTickler documentation

Current version: 1.3.2

**Overview:**

This script is used to make a popup toast notification on the user’s computer reminding to upgrade the OS build. A popup has a “Close” button which closes the popup, but it will be popped as scheduled, and “Install Now” button, which will redirect a user to the SCCM Client to the proper page with the Upgrade button. The popup is scheduled via the Task Scheduler and is executed from a local file without the necessity to be connected to the Houle network. However, notification will be displayed only if a user is connected to the Houle network. For the full list of the IPs, please have a look at the [\\bby-configmgr01\Applications\UserTickler\files\list\_of\_ip.txt](file:///\\bby-configmgr01\Applications\UserTickler\files\list_of_ip.txt)

The notification will be triggered by time, every day after 2pm and is scheduled to run every two hours.

After the upgrade was completed, the script will remove itself from the system (it will clean up the Task Scheduler and the directory it was copied to).

The script is designed to run on the Windows 10 OS.

**Files of the script:**

The script consists of several parts, each with a specific role. The script’s directory is: [\\bby-configmgr01\Applications\UserTickler\](file:///\\bby-configmgr01\Applications\UserTickler\)

1. PushScript.ps1 – the first script to be executed. As it is pushed through the SCCM, it must have a connection to the HE Corp. It copies folder “*files*” to the user’s machine. After that it sets up the permissions recursively on the folder and files. The last step it does is calling the “*files\tickler\_<current\_version>.ps1*” script on the user’s machine so it is executed locally.
2. tickler\_<current\_version>.ps1 – Is a main script with the following hierarchy (more about functions later on):

* main
  + check\_build
    - create\_task
      * check\_ip
        + invoke(“pretty\_popup.exe”)
      * popup\_message

OR

* + - * clean\_up
        + delete\_taskScheduler
        + delete\_folder

1. pretty\_popup.exe – Executable file that is written in PowerShell and is converted for the convenience.
2. UserTickler.xml – XML file that defines the attributes for the Task Scheduler.
3. warning.png – Picture to be used in the notification body
4. list\_of\_ip.txt – List of the IP addresses that will let the notification popup.

**Functions:**

**Note**: the functions principals that are described in here are described as well in the code itself with the proper variable names so it should be fairly easy to understand each function even with a minimal experience in the PowerShell.

**File**: PushScript.ps1

If there are any changes to the file name due to the version control, please change the version variable at the top.

This script will copy an entire folder “*files*” to the user’s computer.

The *setup\_permissions* function will change the permission folder so the Owner of the folder will be “*BUILTIN\Administrators*” with the full control access to the folder. The permissions for the “*BUILTIN\Users*” is removed.

The last line will call for the main script to be executed and since it is executed by an SCCM it will have admin privileges.

**File**: tickler\_<current\_version>.ps1

*main*: calls for the *check\_build*

*check\_build*: required build is 17763 which is the build set up as a current for the Houle at the time this documentation is published. The function checks for the current computer build using the Wmi Class *win32\_operatingsystem*. If it is lower than standard, script will create a task using *create\_task* function. It will check for the current IP using *check\_ip\_address* function and if the computer is connected to the HE Corp will display the notification.

If the current build is up-to-date, the script will initiate a *clean\_up* part.

*create\_task:* The first step is to check if the task has already been registered due to the recursive nature of the script. If it was not registered, the script will register a task called “UserTickler” using pre-defined “UserTickler.xml” XML files.

*check\_ip\_address*: Gets the list of the approved IP addresses and checks the current IP’s subnet. If it the subnet is on the list returns True. Note: the current list contains only /24 (255.255.255.0) subnets. That means it may treat /26 subnets as one /24 subnet and may be a potential problem in the future. The current IP list includes only major offices and places with a fast internet connection. The check is done using a built in class [IPAddress] and comparing properties of the custom object with the subnets.

*popup\_message*: Function simply invokes “pretty\_popup.exe” file.

*clean\_up*: Invokes two functions to unregister and delete the task from the Task Scheduler and to delete the “*C:\UserTickler*” folder recursively.

**File**: pretty\_popup.ps1

MS Documentation links:

* <https://docs.microsoft.com/en-us/uwp/api/windows.ui.notifications.toastnotifier.addtoschedule>
* <https://docs.microsoft.com/en-us/windows/uwp/design/shell/tiles-and-notifications/adaptive-interactive-toasts>

**Note**: This is the source code for the “pretty\_popup.exe” files.

At the beginning it calls for the built in libraries for the UI toast notifications and uses the default template as a template for the popup. *ToastTemplate* variable will be filled with the xml template down in the script.

The popup outlook is formed in the XML, the tags are default and could not be modified, but can be combined to achieve the desired result. It also uses the hardcoded path to the image (it should not be, but it was designed as a script not an Object Oriented Program and could not inherit the variables from other files).

After, the call to the Registries is made to call the PowerShell and execute the code of the toast notifications.

**Restrictions**: Due to the nature of the libraries, the pop up can be executed only by a User or UserGroup that is logged in to the system. It will not be executed on the background or for a user that has not logged in to the system.

**File**: UserTickler.xml

MS Documentation links:

* <https://docs.microsoft.com/en-us/windows/win32/taskschd/using-the-task-scheduler>
* <https://docs.microsoft.com/en-us/windows/win32/taskschd/daily-trigger-example--xml->

This file is used to configure the Task Scheduler. The key point is the <*Principals*> tag as it defines who is going to execute the script. The current setup is set with <*GroupID*> tag and All Domain Users. It will run with the highest privileges as advanced users with admin access can restrict access to some folders.

The <*Triggers*> section is set up to run from 14:00 every day every two hours for the next 100 days.

The <*Exec*> tag defines what is going to be executed and with what parameters. The switch *-ExecutionPolicy ByPassy* specifies to run the script without changing the policy on the executing scripts as by default they are set to restrict any execution.

The <*Author*> tag is required as well as the <*URI*> that is the name of the script in this case to register a task.

**Deployment through the SCCM**

This script is deployed through SCCM as a package.

If any changed in the code are made, you are required to create a new package. Please use a proper name and a description with the modifications or a reference to the documentation.

To create a package:

* Open **Software Library -> Application Management -> Packages** and click **Create Package**.
* Specify the name of the package and set the source folder to the [\\bby-configmgr01\Applications\UserTickler](file:///\\bby-configmgr01\Applications\UserTickler)
* Choose a **Standard Program**
* In the **command line** section, specify the following command to be run: *Powershell.exe -NoProfile -ExecutionPolicy ByPass -File pushScript.ps1*
* Set the **Maximum allowed run time** to 15 min (minimal value)

To deploy the package

* Specify **Collection, Distribution points**.
* The **Purpose** must be Required
* Set the **Assignment schedule** to the “As Soon As Possible” and Rerun behavior “Rerun if failed previous attempt”. Don’t worry, if something goes wrong and it is an unfixed error SCCM will stop trying to rerun the program in couple iterations.
* On the User Experience page, check **Software installation** and uncheck everything else.
* For the **deployment options**, select “Run Program from distribution point” and “Do not run the program”.

You will be able to monitor the deployment of the script as a usual software.