How to Automate PowerShell Scripts

Adam Stetson
Systems Engineer, Security Expert

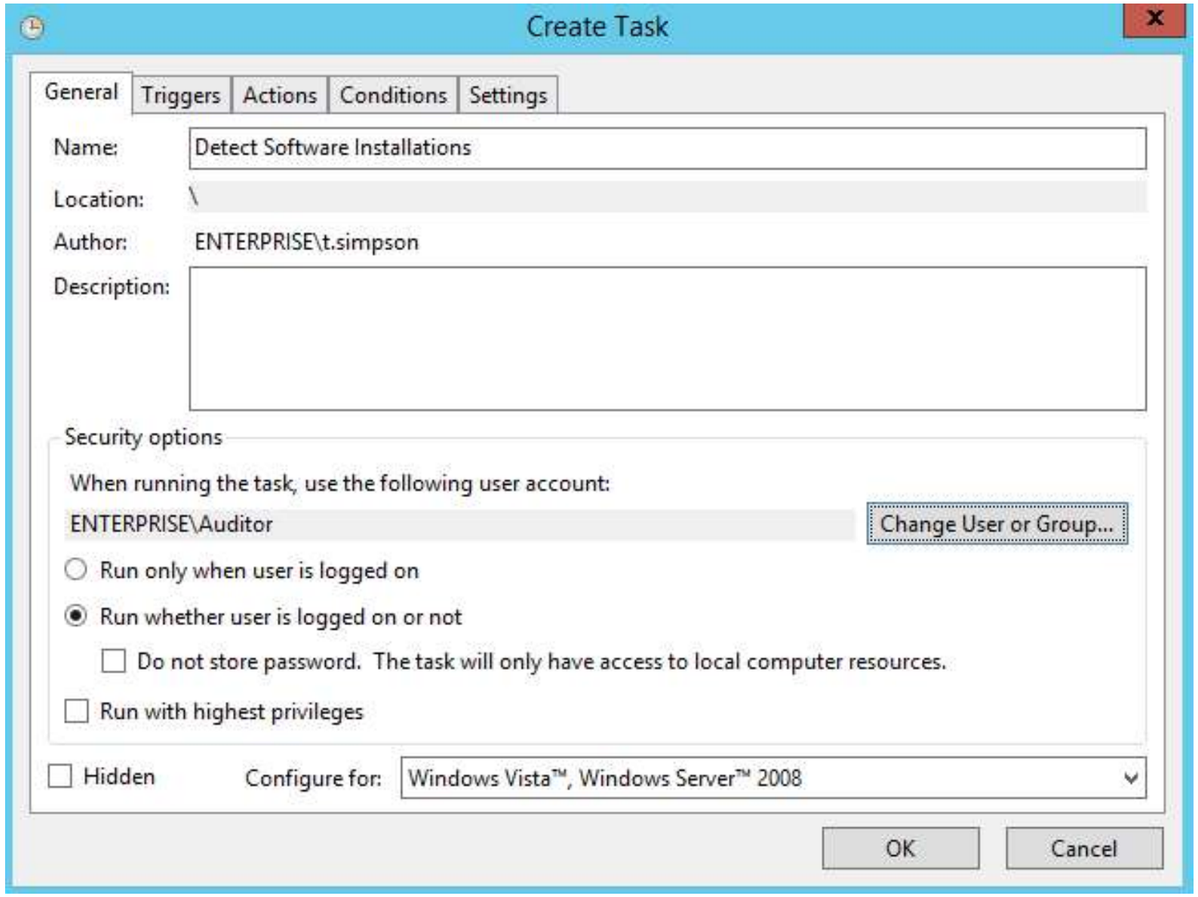
Microsoft Windows Task Scheduler can help you automatically launch a program or PowerShell script at a certain time or when certain conditions are met. You can also schedule sending emails and even displaying certain messages. In this blog, we will show you how to run a PowerShell script from Task Scheduler that will alert on any software installation on a local computer. We will also create scheduled tasks using PowerShell scripts. You will learn how to:

- Create Tasks with Task Scheduler
- Modify or Delete Scheduled Tasks
- Create Scheduled Tasks with PowerShell Scripts.

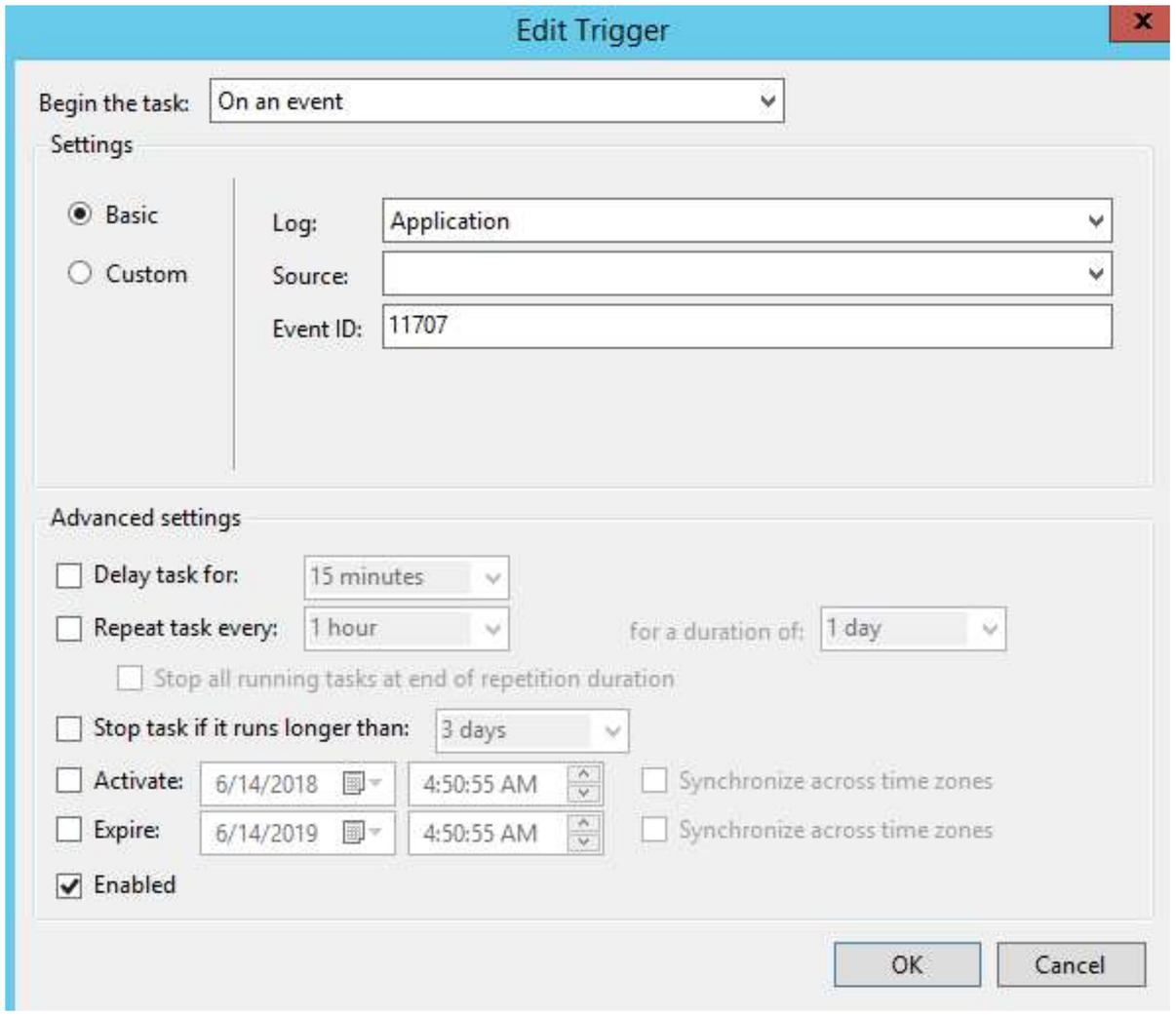
Creating Tasks with Task Scheduler

Open Task Scheduler by pressing "Windows+R" and then typing "taskschd.msc" in the window that opens. Then take the following steps:

1. Click "Create a task" and enter a name and description for the new task. To run the program with administrator privileges, check the "Run with the highest privileges" box. In our example, we'll assign a service account to run the task, and run it regardless of whether the user is logged on.



2. Switch to the Triggers tab and click the "New..." button. Here you can specify the conditions that trigger the task to be executed. For example, you can have it executed on schedule, at logon, on idle, at startup or whenever a particular event occurs. We want our task to be triggered by any new software installation, so we choose "On an event" from the drop-down menu and select "Application" from the Log settings. Leave the "Source" parameter blank and set the EventID to "11707". Click "OK" to save the changes.

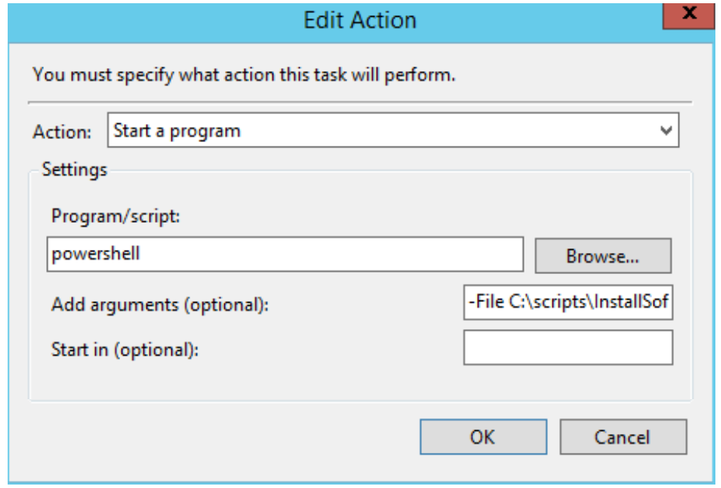


3. Navigate to the "Actions" tab, and click "New...". Here you can specify the actions that will be executed whenever the trigger conditions are met. For instance, you can send an email or display a message. In our case, we want to start a program, so we need to create the PowerShell script we want to run and save it with the "ps1" extension. You can find the script here; it will send an alert with the event details about the installed software.

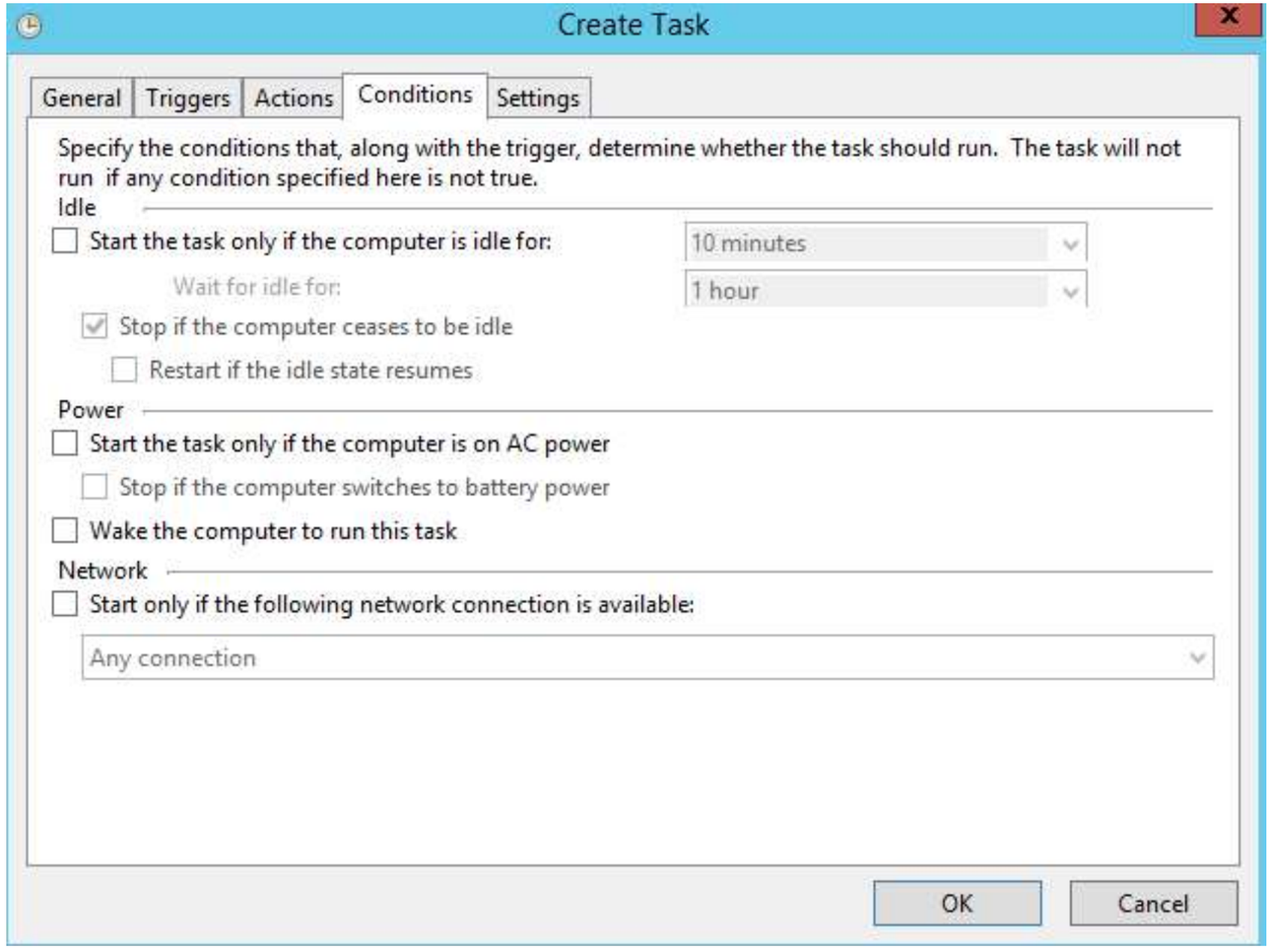
To schedule the PowerShell script, specify the following parameters:

- **Action:** Start a program
- **Program\script:** powershell
- **Add arguments (optional):** -File [Specify the file path to the script here]

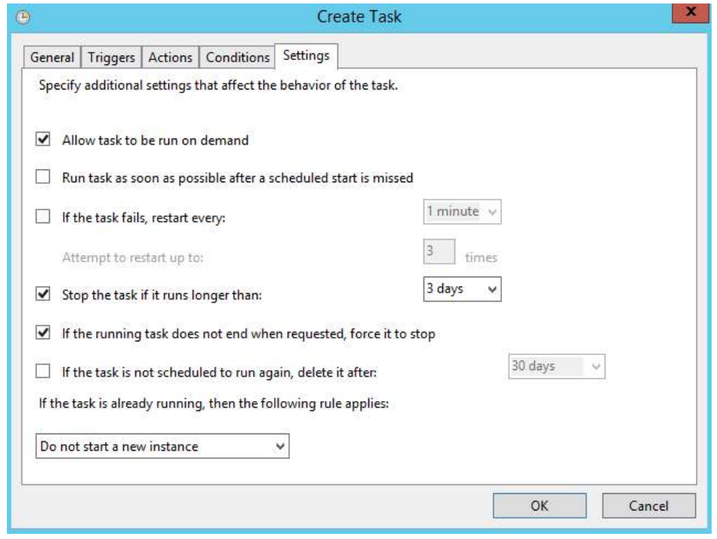
Click "OK" to save your changes.



4. The “Conditions” tab enables you to specify the conditions that, along with the trigger, determine whether the task should be run. In our case, we should leave the default settings on this tab.



5. You can also set up additional parameters for your scheduled task on the “Settings” tab. For our example, though, we’ll leave them unchanged.



6. When the task is completely set up, the system will ask you for the service account password. Note that this account must have the “Log on as Batch Job” right. Enter the password and click “OK” to save the task.

7. For Task Scheduler to function properly, the Job Scheduler service must be set to start Run “Services.msc”. In the list of services, find Task Scheduler and double-click it. On the General tab, set the startup type to “Automatic” and click OK to save your change.

Now whenever new software is installed on your Microsoft Windows Server, you will be notified via an email that details the time of the installation, the name of the software and the user ID (SID) of the person who installed it.

Modifying or Deleting Scheduled Tasks

To modify an existing task, right-click it in the list, select Properties, edit the required settings and click OK. To delete a scheduled task, right-click it, select Delete and confirm the action.

Creating Scheduled Tasks with PowerShell Scripts

Now that you know how to create a task using Task Scheduler, let's find out how to create a scheduled task using PowerShell. Suppose we want our task to be launched daily at 10 AM, and it must execute the PowerShell script you can find here, which will monitor changes to group membership in the Active Directory site.

In Windows Powershell 2.0 (Windows 7, Windows Server 2008 R2), to create a scheduled job, you must use the **TaskScheduler** module. Install the module by running the "**Import-Module TaskScheduler"** command and use the following script to create a task that will execute the PowerShell script named GroupMembershipChanges.ps1 daily at 10 AM:

```
Import-Module TaskScheduler $task = New-Task
$task.Settings.Hidden = $true
Add-TaskAction -Task $task -Path C:\Windows\
system32\WindowsPowerShell\v1.0\powershell.
exe -Arguments "-File C:\Scripts\GroupMem-
bershipChanges.ps1"
Add-TaskTrigger -Task $task -Daily -At
"10:00"
Register-ScheduledJob -Name "Monitor Group
Management" -Task $task
```

Windows PowerShell 4.0 (Windows Server 2012 R2 and above) doesn't include the Task Scheduler module, so this script will not work. Instead, PowerShell 3.0 and 4.0 introduced new cmdlets for creating scheduled tasks, **New-ScheduledTaskTrigger** and **Register-ScheduledTask**, which make creating a scheduled task much easier and more convenient. So let's create a task that will execute our script daily at 10 AM using the system account (SYSTEM). This task will be performed by an account with elevated privileges.

```
Import-Module TaskScheduler $task = New-Task
$task.Settings.Hidden = $true
Add-TaskAction -Task $task -Path C:\Windows\
system32\WindowsPowerShell\v1.0\powershell.exe
-Arguments "-File C:\Scripts\GroupMembership-
Changes.ps1"
Add-TaskTrigger -Task $task -Daily -At "10:00"
Register-ScheduledJob -Name "Monitor Group Ma-
nagement" -Task $task
```

```
PS C:\Windows\system32> $Trigger= New-ScheduledTaskTrigger -At 10:00am -Daily # Specify the t
$User= "NT AUTHORITY\SYSTEM" # Specify the account to run the script
$Action= New-ScheduledTaskAction -Execute "PowerShell.exe" -Argument "C:\PS\StartupScript.ps1
Register-ScheduledTask -TaskName "MonitorGroupMembership" -Trigger $Trigger -User $User -Acti

TaskPath                                    TaskName                          State
--------                                    --------                          -----
\                                           MonitorGroupMembership            Ready
```

Other trigger options that could be useful in creating new tasks include:

- **-AtStartup** — Triggers your task at Windows startup.
- **-AtLogon** — Triggers your task when the user signs in.
- **-Once** — Triggers your task once. You can set a repetition interval using the –RepetitionInterval parameter.
- **-Weekly** — Triggers your task once a week.

Note that, using these cmdlets, it is not possible to trigger execution “on an event” as we did with the Task Scheduler tool. PowerShell scripts with “on an event” triggers are much more complicated, so this is a real disadvantage of using PowerShell rather than Task Scheduler.

As you can see, it is easy to create scheduled tasks using Task Scheduler or PowerShell. But remember that improper changes to your scheduled tasks can cause service interruptions and degrade server performance. Therefore, it’s essential to track all changes to your scheduled tasks.