- 1. 4. 2. Automating tasks and activities -

Using Task Scheduler

Task Scheduler is a Microsoft Management Console (MMC) snap-in that supports an extensive set of triggering and scheduling options.

You can run programs or scripts at specified times, launch actions when a computer has been idle for a specified period of time, run tasks when particular users sign in or out, and so on.

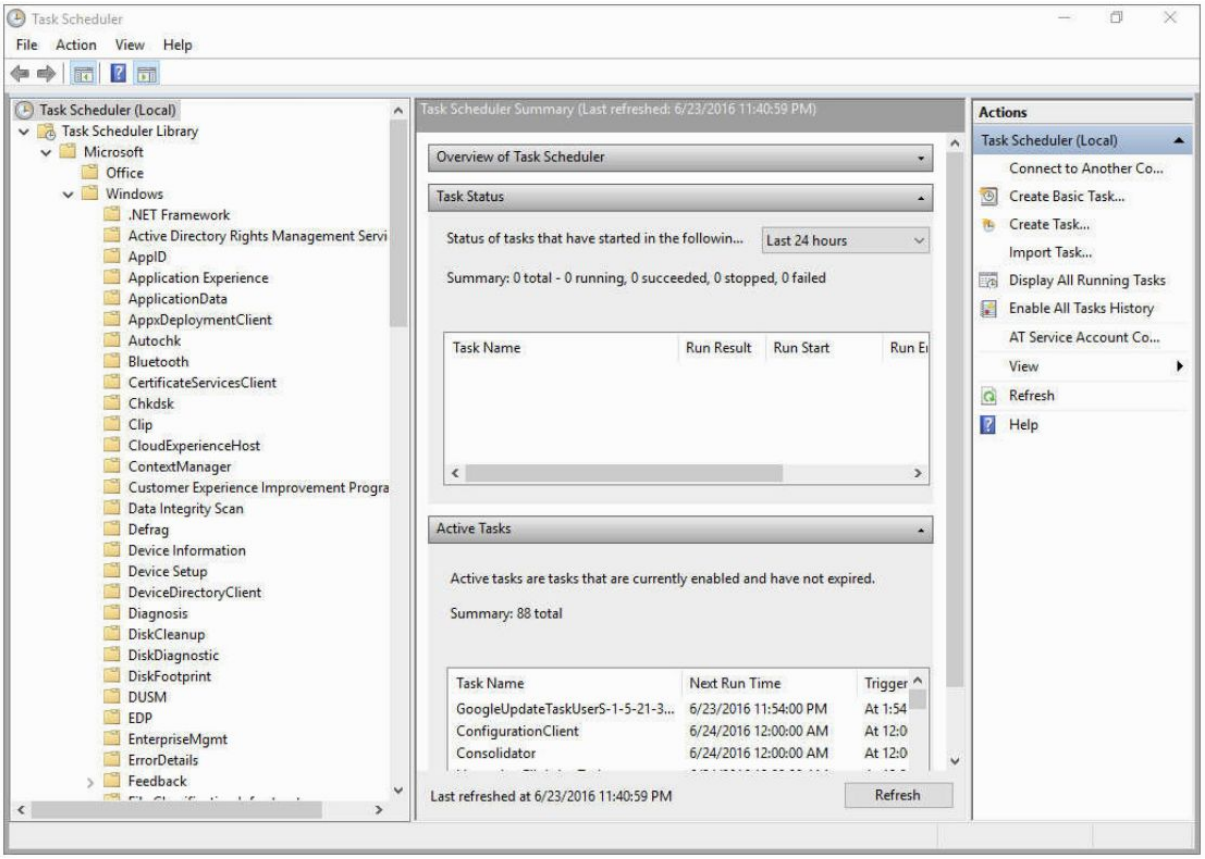
Task Scheduler is also tightly integrated with the Event Viewer snap-in, making it easy for you to use events (an application crash or a disk-full error, for example) as triggers for tasks.

To launch Task Scheduler, in the search box, type "sched" .

In the results list, click Task Scheduler.

Alternatively, press Windows key+R and type taskschd.msc in the Run box.

The next figure shows a sample of Task Scheduler in its default layout:



As you can see, the window is divided into three regions—a console tree on the left, the Actions pane on the right, and, in between, various informative windows in the details pane.

The console tree shows you which computer you're working with (the local machine or a network computer) and provides a folder tree of currently defined tasks.

You can create your own folders here to organize the tasks you create yourself, or you can add new tasks to existing folders.

The Actions pane provides a menu of things you can do.

Some, but not all, of the items here are also available on the menus at the top of the window.

If your screen is too crowded with the Actions pane displayed, you can hide—and redisplay—it using the button at the right end of the toolbar, directly below the menu bar.

In the center part of the window, initially you see an overview message (which is a bit of static text you can hide by clicking the collapse arrow at the right), a status report of all tasks that have run (or were scheduled to run) during some period of time (by default, the most recent 24 hours), and a summary of all the currently enabled tasks.

Entries in the Task Status list have outline controls; click an item's plus sign to see more details.

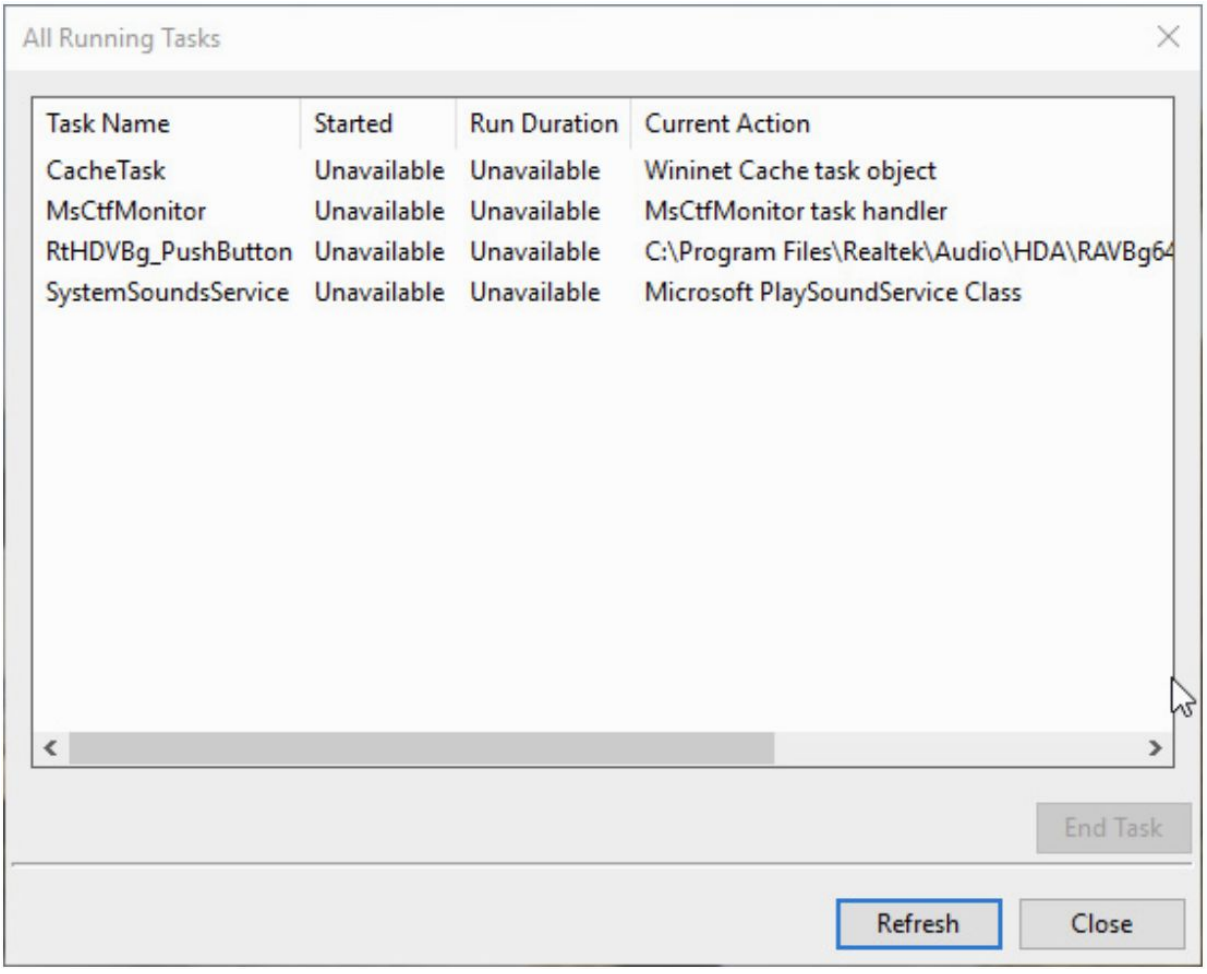
The Task Status and Active Tasks areas are not updated automatically.

To get the latest information, click Refresh at the bottom of the screen, in the Actions pane, or on the Action menu.

If this is your first visit to Task Scheduler, you might be surprised by the number of active tasks that Windows and your applications have already established.

Windows and third-party apps make extensive use of Task Scheduler to set up maintenance activities that run on various schedules.

You can see which tasks managed by Task Scheduler are currently running by clicking Display All Running Tasks in the Actions pane.



To satisfy your curiosity about what an active task does and how it has been set up, you need to locate it in the console tree.

Expand the outline entries as needed, and browse to an item of interest.

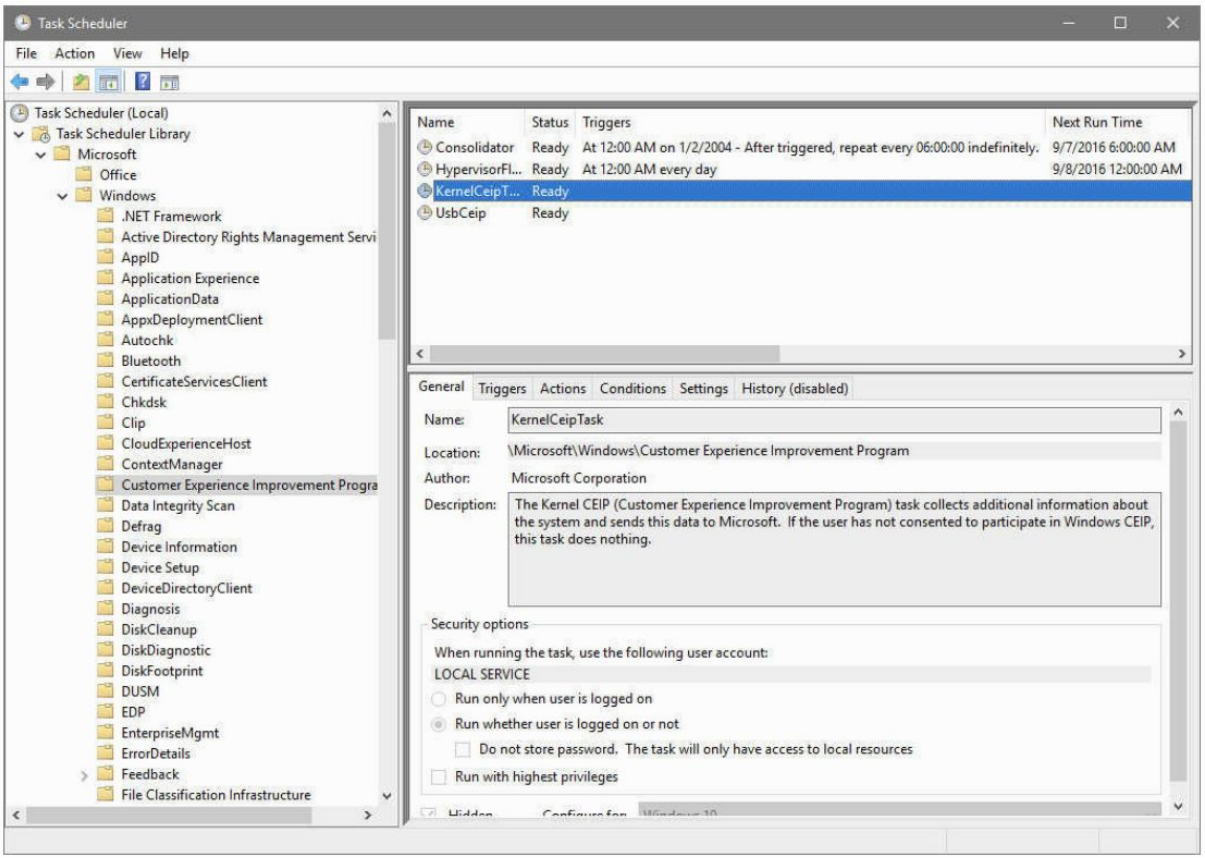
The entries in the console tree are virtual folders, each of which can contain subfolders or one or more tasks.

When you select a folder, the upper part of the details pane lists all tasks stored in the folder.

The lower area of the pane, meanwhile, shows a tabbed display of the properties of the selected task.

The following picture shows the Customer Experience Improvement Program folder selected in the console tree, the task named KernelCeipTask selected in the upper area of the details pane, and the General tab of the KernelCeipTask properties displayed in the lower area.

The Actions pane has been hidden in this figure.



The properties display that appears is read-only.

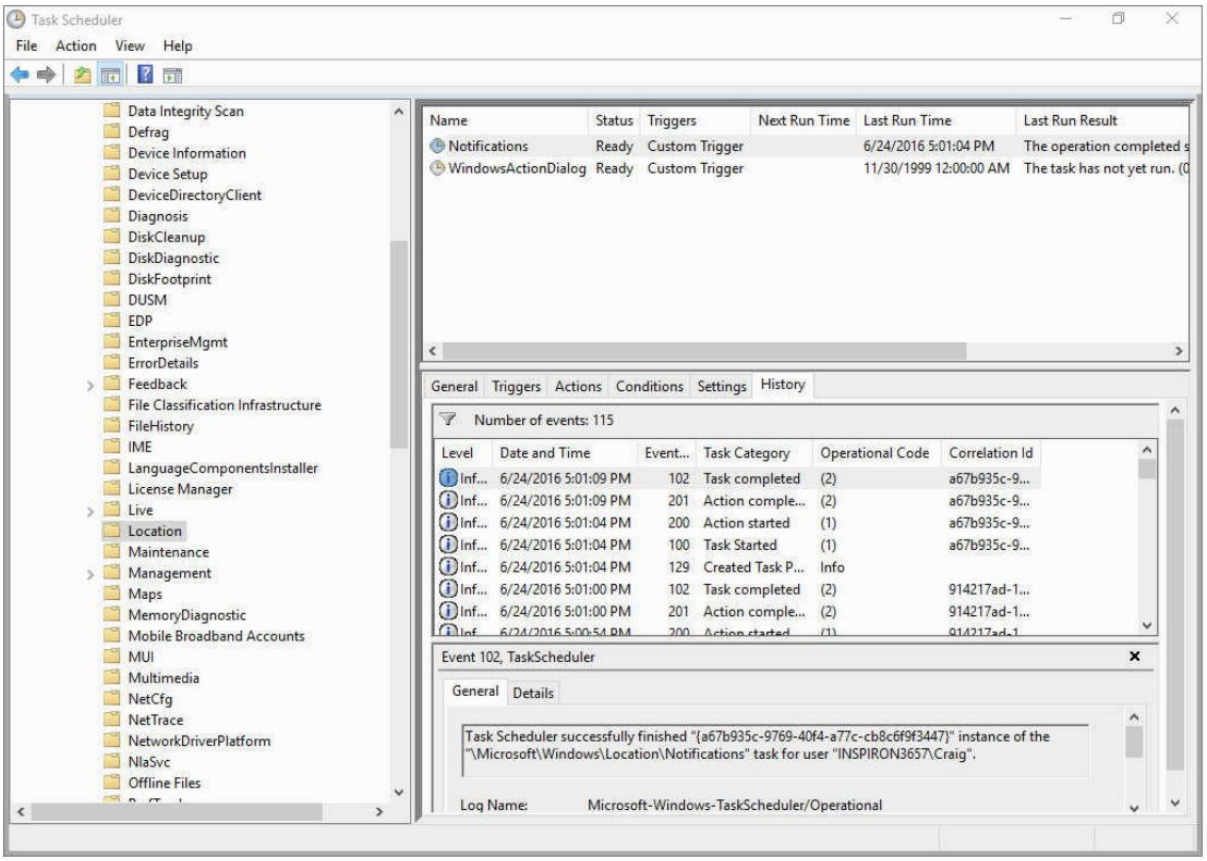
To edit the properties associated with a task, right-click the task name and then click Properties (or double-click the task's entry).

That will open a read/write dialog box in a separate window.

With the exception of the History tab, the properties dialog box is identical to the Create Task dialog box, one of the tools you can use to create a new task; we'll explore that dialog box in some detail in the following section, "Creating a task."

On the History tab, you can see exactly how, whether, and when a task has run.

The next image shows the History tab for the Notifications task in the Microsoft\Windows\Location folder.



If the History tab is disabled, click Enable All Tasks History in the Actions pane.

When you display the History tab, the relevant portion of the Event Viewer snap-in snaps in, showing you all the recent events related to the selected task.

This is exactly what you see if you run Eventvwr.msc, navigate in the console tree to Applications And Services Logs\Microsoft\Windows\TaskScheduler\Operational, and filter the resulting log to show events related to the selected task.

Obviously, if you want this information, it's quicker to find it in the Task Scheduler console than in the Event Viewer console.

If a task you set up is not being triggered when you expect it to or not running successfully when it should, you can double-click the appropriate event entry and read whatever details the event log has to offer.

The Windows 10 Task Scheduler maintains an ample history of the events generated by each task.

If a task is failing regularly or intermittently, you can review all the causes by scrolling through the History tab on the task's properties display.

Task Scheduler terminology

As you go through the steps to create or edit a task, you'll encounter the following terms:

- Trigger. The time at which a task is scheduled to run or the event in response to which a task runs. A task can have multiple triggers.
- Action. What the task does. Possible actions include starting a program, sending an email message, and displaying a message on the screen. A task can have multiple actions, in which case the actions occur sequentially in the order in which you assign them.
- Condition. An additional requirement that, along with the trigger, must be met for the task to run. For example, a condition might stipulate that the task run only if the computer has been idle for 10 minutes or only if it's running on AC power.
- Setting. A property that affects the behavior of a task. With settings, you can do such things as enable a task to run on demand or set retry parameters to be followed if a task fails to run when it's triggered.

Creating a task

To begin creating a new task, select the folder in the console tree where you want the task to reside.

If you want to create a new folder for this purpose, right-click the folder's parent in the console tree and click New Folder.

You can create a new task in the Task Scheduler snap-in by using a wizard or by filling out the Create Task dialog box.

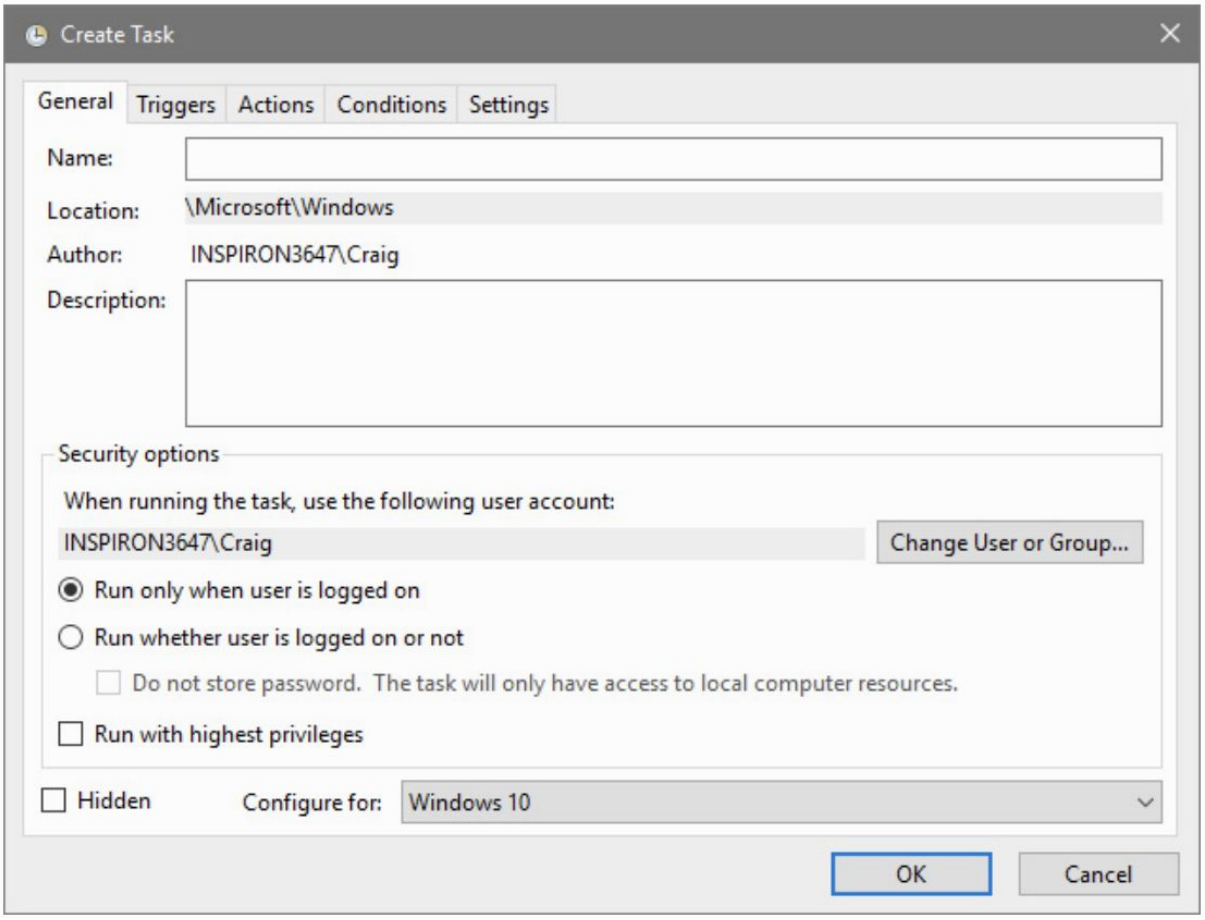
The wizard, which you launch by clicking Create Basic Task (in the Actions pane or on the Action menu), is ideal for time-triggered tasks involving a single action.

It's also fine for setting up a task to run when you sign in or when Windows starts.

For a more complex task definition, you need to work through the Create Task dialog box.

Select the folder where you want the task to appear (in the console tree), and then click Create Task in the Actions pane or on the Action menu.

The next figure shows the General tab of the Create Task dialog box:



The one required entry on the General tab is a name for the task; everything else is optional.

The task's author is you (you can't change that), and unless you specify otherwise, the task will run in your own security context.

If you want it to run in the security context of a different user or group, click Change User Or Group and fill out the ensuing dialog box.

The circumstance under which you're most likely to need to change the security context is when you're setting up tasks to run on another computer.

If you intend to run programs with which another user can interact, you should run those in the other user's security context.

If you run them in your own, the tasks will run noninteractively (that is, the user will not see them).

Regardless of which user's security context the task is to run in, you have the option of allowing the task to run whether that user is signed in or not.

If you select Run Whether User Is Logged On Or Not, you will be prompted for the user's password when you finish creating the task.

If you don't happen to have that password, you can select Do Not Store Password.

As the text beside this check box indicates, the task will have access to local resources only.

Setting up a task's trigger or triggers

Tasks can be triggered in the following ways:

- On a schedule.
- At logon.
- At startup.
- On idle.
- On an event.
- At task creation or modification.
- On connection to a user session.
- On disconnection from a user session.
- On workstation lock.
- On workstation unlock.

You can establish zero, one, or several triggers for a task.

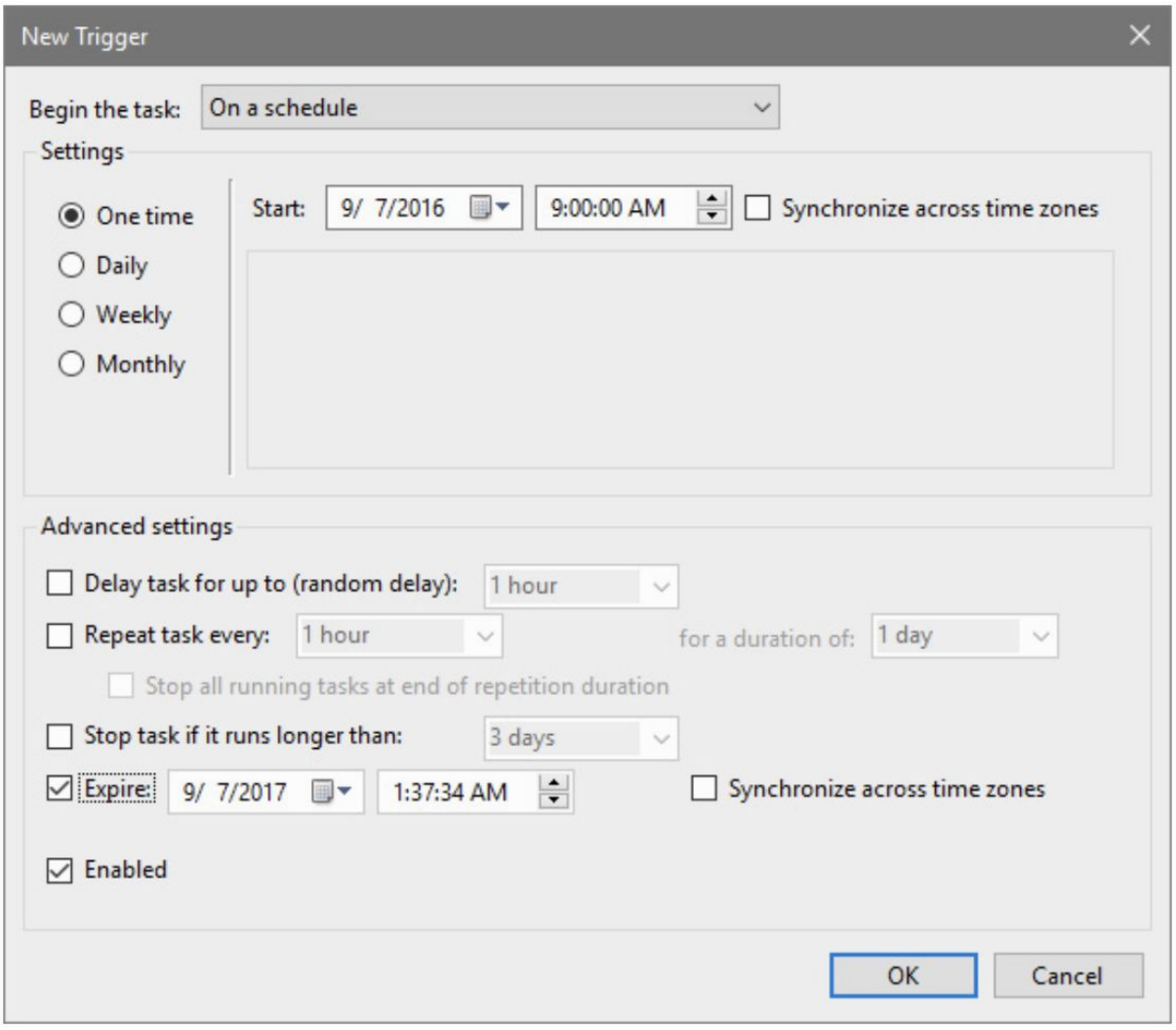
If you don't set any triggers, you can still run the task on demand (unless you clear the Allow Task To Be Run On Demand check box on the Settings tab of the Create Task dialog box).

Running the task on demand gives you a way to test a new task before committing it to a schedule, for example.

If you set multiple triggers, the task runs when any one of the triggers occurs.

To set up a trigger, click the Triggers tab in the Create Task dialog box, and then click New.

In the New Trigger dialog box (shown in the next figure), choose the type of trigger you want from the Begin The Task list.



Note the Advanced Settings options at the bottom of the dialog box shown in the previous figure.

These choices—which you use to establish delay, repeat, and expiration parameters (among other things)—are not so easy to find when you're reviewing a task that you or someone else has already created.

They don't appear in the read-only version of a task's properties, and in the read/write version of the properties dialog box, you need to select a trigger (on the Triggers tab) and click Edit to see or change the advanced settings.

Setting up a task's action or actions

In addition to the task name (which you supply on the General tab of the Create Task dialog box), the only other task parameter you must provide is the action or actions the task is supposed to perform.

This you do by clicking New on the Actions tab and filling out the rest of the dialog box.

Opening the Start A Program drop-down menu, you will find three choices: Start A Program, Send An E-Mail, and Display A Message.

The second and third of these, however, have been deprecated since Windows 8.

So leave Action set at Start A Program, and then supply the program (or script) name, optional arguments, and an optional start location.

You can specify one or several actions.

Multiple actions are carried out sequentially, with each new action beginning when the previous one is complete.

The Start A Program option can be applied to anything Windows can execute—a Windows program, a batch program or script, a document associated with a program, or a shortcut.

You can use the Browse button to simplify entry of long path specifications, add command-line parameters for your executable on the Add Arguments line, and specify a start-in folder for the executable.

If your program needs elevated privileges to run successfully, be sure you select Run With Highest Privileges on the General tab of the Create Task dialog box.

Automating command sequences with batch programs

A batch program (also commonly called a batch file) is a text file that contains a sequence of commands to be executed.

You execute the commands by entering the file name at a command prompt.

Any action you can take by typing a command at a command prompt can be encapsulated in a batch program.

When you type the name of your batch program at the command prompt (or when you specify it as a task to be executed by Task Scheduler and the appropriate trigger occurs), the command interpreter opens the file and starts reading the statements.

It reads the first line, executes the command, and then goes on to the next line.

On the surface, this seems to operate just as though you were typing each line yourself at the command prompt.

In fact, however, the batch program can be more complicated because the language includes replaceable parameters, conditional and branching statements, the ability to call subroutines, and so on.

Batch programs can also respond to values returned by programs and to the values of environment variables.

Working at the command prompt

To get to the command prompt, run Cmd.exe.

You can do this by double-clicking any shortcut for Cmd.exe, but because you like to type, you might find it easiest to press Windows key+R, and then type "cmd".

Running with elevated privileges

Your activities in a Command Prompt session are subject to the same User Account Control (UAC) restrictions as anything else you do in Windows.

If you use Command Prompt to launch a program (for example, Registry Editor) that requires an administrative token, you'll be asked to confirm a UAC prompt before moving on.

If you plan to run several such tasks from Command Prompt, you might prefer to run Cmd.exe itself with elevated privileges.

To do this, right-click any shortcut for Command Prompt and then click Run As Administrator.

Or right-click the Start button and click Command Prompt (Admin).

Windows displays the word Administrator in the title bar of any Command Prompt window running with elevated privileges.

Editing the command line

When working at a command prompt, you often enter the same command multiple times or enter several similar commands.

To assist you with repetitive or corrective tasks, Windows includes a feature that recalls previous commands and allows you to edit them on the current command line:

Key	Function
Up Arrow or F3	Recalls the previous command in the command history
Down Arrow	Recalls the next command in the command history
Page Up	Recalls the earliest command used in the session
Page Down	Recalls the most recently used command
Left Arrow	Moves left one character
Right Arrow	Moves right one character
Ctrl+Left Arrow	Moves left one word
Ctrl+Right Arrow	Moves right one word
Home	Moves to the beginning of the line
End	Moves to the end of the line
Esc	Clears the current command
F7	Displays the command history in a scrollable pop-up box
F8	Displays commands that start with the characters currently on the command line
Alt+F7	Clears the command history

Using command symbols

Old-fashioned programs that take all their input from a command line and then run unaided can be useful in a multitasking environment.

You can turn them loose to perform complicated processing in the background while you continue to work with other programs in the foreground.

To work better with other programs, many command-line programs follow a set of conventions that control their interaction:

- By default, programs take all of their input as lines of text typed at the keyboard. But input in the same format also can be redirected from a file or any device capable of sending lines of text.
- By default, programs send all of their output to the screen as lines of text. But output in the same format also can be redirected to a file or another line-oriented device, such as a printer.
- Programs set a number (called a return value) when they terminate to indicate the results of the program.

When programs are written according to these rules, you can use the symbols listed in the following table to control a program's input and output or chain programs together:

Symbol	Function
<	Redirects input
>	Redirects output
>>	Appends redirected output to existing data
|	Pipes output
&	Separates multiple commands in a command line
&&	Runs the command after && only if the command before && is successful
||	Runs the command after || only if the command before || fails
^	Treats the next symbol as a character
(and)	Groups commands

The redirection symbols

Command Prompt sessions in Windows allow you to override the default source for input (the keyboard) or the default destination for output (the screen).

Redirecting output

To redirect output to a file, type the command followed by a greater-than sign (>) and the name of the file.

Using two greater-than signs (>>) redirects output and appends it to an existing file.

Redirecting input

To redirect input from a file, type the command followed by a less-than sign (<) and the name of the file.

Redirecting input and output

You can redirect both input and output in a command line.

For example, to use Batch.lst as input to the Sort command and send its output to a file named Sorted.lst, type the following:

Sort < batch.lst > sorted.lst

Standard output and standard error

Programs can be written to send their output either to the standard output device or to the standard error device.

Sometimes programs are written to send different types of output to each device.

You can't always tell which is which because, by default, both devices are the screen.

The Type command illustrates the difference.

When used with wildcards, the Type command sends the name of each matching file to the standard error device and sends the contents of the file to the standard output device.

Because they both go to the screen, you see a nice display with each file name followed by its contents.

However, if you try to redirect output to a file by typing something like this:

type *.bat > std.out

the file names still appear on your screen because standard error is still directed to the screen.

Only the file contents are redirected to Std.out.

With Windows, you can qualify the redirection symbol by preceding it with a number.

Use 1> (or simply >) for standard output and 2> for standard error.

For example:

type *.bat 2> err.out

This time the file contents go to the screen and the names are redirected to Err.out.

The pipe symbol

The pipe symbol (|) is used to send, or pipe, the output of one program to a second program as the second program's input.

Piping is commonly used with the More command, which displays multiple screenfuls of output one screenful at a time.

For example:

help dir | more

This command line uses the output of Help as the input for More.

The More command filters out the first screenful of Help output, sends it to the screen as its own output, and then waits for a keystroke before sending more filtered output.

Pipe command-line output to the Clipboard

Using the Clip utility, introduced with Windows Vista, you can pipe the output of a command to the Windows Clipboard, from whence you can paste it into any program that accepts Clipboard text.

Typing **dir | clip** , for example, puts a listing of the current directory's files on the Clipboard.

You can also redirect the contents of a file to the Clipboard using the < symbol.

Typing **clip < myfile.txt** , for example, transfers the contents of myfile.txt to the Clipboard.

- Vocabulary -

- idle: parado / en reposo / sin actividad.
- behaviour: comportamiento.

- Exercises - 1. 4. 2. Automating tasks and activities -

Open the following Google Document that you have created in a previous sub-unit:

"1. 4. Windows 10 for experts and IT pros - Apellidos, Nombre"

being "Apellidos, Nombre" your Last Name and Name.

Inside this Google Document you are going to copy and answer all the "Exercises" of this sub-unit:

1. Go to the Windows search box, type “sched” and in the results list, click on “Task Scheduler”. To create a new task, select the “Task Scheduler Library” folder in the console tree and create a new folder named “DAM”. Go to the “Action” menu and select “Create Task”. In the “General” tab: “Name” -> “SI Task”, “Security Options” -> “Run only when user is logged on”. In the “Triggers” tab: “New” -> “Begin the task” -> “At log on”. In the “Actions” tab: “New” -> “Action” -> “Start a program” -> “Program/script:” C:\Windows\System32\calc.exe . When you will restart your computer, the Windows Calculator should start automatically.
2. What do the following CMD commands make? cd, chkdsk, cls, copy, del, dir, echo, exit, find, help, ipconfig, mkdir, more, net, net use, netstat, pause, ping, rename, rmdir, shutdown, sort, tracert, tree, whoami.