# Editing textual runbooks in Azure Automation

The textual editor in Azure Automation can be used to edit [PowerShell runbooks](/documentation/articles/automation-runbook-types#powershell-runbooks) and [PowerShell Workflow runbooks](/documentation/articles/automation-runbook-types#powershell-workflow-runbooks). This has the typical features of other code editors such as intellisense and color coding with additional special features to assist you in accessing resources common to runbooks. This article provides detailed steps for performing different functions with this editor.

The textual editor includes a feature to insert code for activities, assets, and child runbooks into a runbook. Rather than typing in the code yourself, you can select from a list of available resources and have the appropriate code inserted into the runbook.

Each runbook in Azure Automation has two versions, Draft and Published. You edit the Draft version of the workflow and then publish it so it can be executed. The Published version cannot be edited. See [Publishing a runbook](/documentation/articles/automation-creating-importing-runbook#publishing-a-runbook) for more information.

To work with [Graphical Runbooks](/documentation/articles/automation-runbook-types#graphical-runbooks), see [Graphical authoring in Azure Automation](/documentation/articles/automation-graphical-authoring-intro).

## To edit a runbook with the Azure Management Portal

Use the following procedure to open a runbook for editing in the textual editor.

1. In the Azure Management Portal, select your automation account.
2. Click the **Runbooks** tile to open the list of runbooks.
3. Click the name of the runbook you want to edit and then click the **Edit** button.
4. Perform the required editing.
5. Click **Save** when your edits are complete.
6. Click **Publish** if you want the latest draft version of the runbook to be published.

### To insert a cmdlet into a runbook

1. In the Canvas of the textual editor, position the cursor where you want to place the cmdlet.
2. Expand the **Cmdlets** node in the Library control.
3. Expand the module containing the cmdlet you want to use.
4. Right click the cmdlet to insert and select **Add to canvas**. If the cmdlet has more than one parameter set, then the default set will be added. You can also expand the cmdlet to select a different parameter set.
5. The code for the cmdlet is inserted with its entire list of parameters.
6. Provide an appropriate value in place of the data type surrounded by braces <> for any required parameters. Remove any parameters you don’t need.

### To insert code for a child runbook into a runbook

1. In the Canvas of the textual editor, position the cursor where you want to place the code for the [child runbook](/documentation/articles/automation-child-runbooks).
2. Expand the **Runbooks** node in the Library control.
3. Right click the runbook to insert and select **Add to canvas**.
4. The code for the child runbook is inserted with any placeholders for any runbook parameters.
5. Replace the placeholders with appropriate values for each parameter.

### To insert an asset into a runbook

1. In the Canvas of the textual editor, position the cursor where you want to place the code for the child runbook.
2. Expand the **Assets** node in the Library control.
3. Expand the node for the type of asset you want.
4. Right click the asset to insert and select **Add to canvas**. For [variable assets](/documentation/articles/variable-assets), select either **Add “Get Variable” to canvas** or **Add “Set Variable” to canvas** depending on whether you want to get or set the variable.
5. The code for the asset is inserted into the runbook.

## To edit a runbook with the Azure Management Portal

Use the following procedure to open a runbook for editing in the textual editor.

1. In the Azure Management Portal, select **Automation** and then then click the name of an automation account.
2. Select the **Runbooks** tab.
3. Click the name of the runbook you want to edit and then select the **Author** tab.
4. Click the **Edit** button at the bottom of the screen.
5. Perform the required editing.
6. Click **Save** when your edits are complete.
7. Click **Publish** if you want the latest draft version of the runbook to be published.

### To insert an activity into a Runbook

1. In the Canvas of the textual editor, position the cursor where you want to place the activity.
2. At the bottom of the screen, click **Insert** and then **Activity**.
3. In the **Integration Module** column, select the module that contains the activity.
4. In the **Activity** pane, select an activity.
5. In the **Description** column, note the description of the activity. Optionally, you can click View detailed help to launch help for the activity in the browser.
6. Click the right arrow. If the activity has parameters, they will be listed for your information.
7. Click the check button. Code to run the activity will be inserted into the runbook.
8. If the activity requires parameters, provide an appropriate value in place of the data type surrounded by braces <>.

### To insert code for a child runbook into a runbook

1. In the Canvas of the textual editor, position the cursor where you want to place the [child runbook](/documentation/articles/automation-child-runbooks).
2. At the bottom of the screen, click **Insert** and then **Runbook**.
3. Select the runbook to insert from the center column and click the right arrow.
4. If the runbook has parameters, they will be listed for your information.
5. Click the check button. Code to run the selected runbook will be inserted into the current runbook.
6. If the runbook requires parameters, provide an appropriate value in place of the data type surrounded by braces <>.

### To insert an asset into a runbook

1. In the Canvas of the textual editor, position the cursor where you want to place the activity to retrieve the asset.
2. At the bottom of the screen, click **Insert** and then **Setting**.
3. In the **Setting Action** column, select the action that you want.
4. Select from the available assets in the center column.
5. Click the check button. Code to get or set the asset will be inserted into the runbook.

## To edit an Azure Automation runbook using Windows PowerShell

To edit a runbook with Windows PowerShell, you edit the workflow using the editor of your choice and save it to a .ps1 file. You can use the [Get-AzureAutomationRunbookDefinition](http://aka.ms/runbookauthor/cmdlet/getazurerunbookdefinition) cmdlet to retrieve the contents of the runbook and then [Set-AzureAutomationRunbookDefinition](http://aka.ms/runbookauthor/cmdlet/setazurerunbookdefinition) cmdlet to replace the existing draft workflow with the modified one.

### To Retrieve the Contents of a Runbook Using Windows PowerShell

The following sample commands show how to retrieve the script for a runbook and save it to a script file. In this example, the Draft version is retrieved. It is also possible to retrieve the Published version of the runbook although this version cannot be changed.

$automationAccountName = "MyAutomationAccount"  
$runbookName = "Sample-TestRunbook"  
$scriptPath = "c:\runbooks\Sample-TestRunbook.ps1"  
  
$runbookDefinition = Get-AzureAutomationRunbookDefinition -AutomationAccountName $automationAccountName -Name $runbookName -Slot Draft  
$runbookContent = $runbookDefinition.Content  
  
Out-File -InputObject $runbookContent -FilePath $scriptPath

### To Change the Contents of a Runbook Using Windows PowerShell

The following sample commands show how to replace the existing contents of a runbook with the contents of a script file containing a workflow. Note that this is the same sample procedure as in [To import a runbook from a script file with Windows PowerShell](/documentation/articles/automation-creating-or-importing-a-runbook#ImportRunbookScriptPS).

$automationAccountName = "MyAutomationAccount"  
$runbookName = "Sample-TestRunbook"  
$scriptPath = "c:\runbooks\Sample-TestRunbook.ps1"  
  
Set-AzureAutomationRunbookDefinition -AutomationAccountName $automationAccountName -Name $runbookName -Path $scriptPath -Overwrite  
Publish-AzureAutomationRunbook –AutomationAccountName $automationAccountName –Name $runbookName

## Related articles

* [Creating or importing a runbook in Azure Automation](/documentation/articles/automation-creatnig-importing-runbook)
* [Learning PowerShell workflow](/documentation/articles/automation-powershell-workflow)
* [Certificates](/documentation/articles/automation-certificates)
* [Connections](/documentation/articles/automation-connections)
* [Credentials](/documentation/articles/automation-credentials)
* [Schedules](/documentation/articles/automation-schedules)
* [Variables](/documentation/articles/automation-variables)