

# How to Runbook

This guide is a quick manual for an Azure Runbook. Here is a brief description of the prerequisites and definitions that must be met in order to create an automation via Azure Runbooks.

## Prerequisites

### Defined goal

A defined goal is necessary that we can check if our end product is doing what we are expecting. This can be as simple as one phrase.

#### Example:

My Azure Runbook should start by a manual trigger and create a new enrollment service profile via Graph API in Intune. As a result, it should return a success-code and the name and id of the new enrollment service profile.

### Defined input

To know how to start the code, we first need to define the input(s) we need to get with the trigger to reach our goal. This must be a static text input.

#### Example:

- Variable 1: \$ServiceProfileName = <textinput>
- Variable 2: \$GroupId = <idinput>
- Variable 3: \$CompanyName = <companyname>

### Defined output

The output should include the name and the id of the created enrollment service profile and if the script was run successful.

#### Example:

```
ESP-WIN-LNC-DEVICE-CompanyEnrollment-PROD // 396e26cb-d657-4a4e-8beb-105ba1984c63
```

```
Script finished successfully
```

# Trigger method

The method is important that we know what order which object(s) are triggering this runbook. This could be a manual user via the Azure Portal or via the Azure Management API. Furthermore, it could be a third-party system, like ServiceNow or a schedule.

## Example:

This Azure Runbook should get triggered manually via the Azure Portal.

# Logic model

The logic model defines the logic of the code in a very high-level draft.

## Example:

1. Get Input
2. Check input values
3. Create JSON body for Graph API
4. Post JSON body to Graph API
5. Get response from Graph API
6. Write response in output

# Decisions

## Coding language

Azure Runbooks can execute different code with different languages. This is defined per Azure Runbook. To change the Runbooks coding language, you can create a new Runbook in the Automation Account. Azure Runbooks offers these languages for code execution:

- PowerShell
- Python

## Runtime

On which environment should the Runbook run on? You can choose between Azure managed hosts or hybrid worker. Hybrid worker brings the advantage that you can connect to on premise resources and bypass some restrictions which can occur if Azure is selected as runtime.


# Procedure

## Create Automation Account

Automation Accounts are the container which contains all the information, secret, Runbook and more. You can create one Automation Account for all your Runbooks. However, it is a good idea if the automation accounts are subdivided according to the technologies that will be automated.

## Create runbook

The runbook contains the code which will be executed. On creation you have to select which type it should create. This is a selection of the mentioned coding languages.

 **Create a runbook** ...

Name \* ⓘ

Runbook type \* ⓘ

Select the type of runbook you want to create:

- **PowerShell runbooks** are text runbooks based on PowerShell.
- **Graphical runbooks** are based on Windows PowerShell and are created and edited completely in the Automation graphical editor.
- **Python runbooks** are text runbooks based on Python scripts targeting the Python interpreter.
- **PowerShell Workflow runbooks** are text runbooks based on Windows PowerShell Workflow.
- **Graphical PowerShell Workflow runbooks** are based on Windows PowerShell Workflow and are created and edited completely in the Automation graphical editor. Runbooks of this type can be migrated to Graphical runbooks.

## Code

When created you can copy your code into the web editor of the Function or start to develop the code directly in the Azure Portal.

## Testing

For testing you don't have to leave the Azure Portal. While developing you are able to activate the "Test pane". There you can enter the variables if parameters are defined on the top level in the code.

Save Publish Revert to published **Test pane** Feedback

```
> CMDLETS
> RUNBOOKS
> ASSETS
```

```
1 $CredentialObject=Get-AutomationPSCredential -Name 'CRED-RB-ALL-CRED-DCDiag-PROD-WE'
2 # $Password = $CredentialObject.GetNetworkcredential().password
3 $Username = $CredentialObject.GetNetworkcredential().username
4
5 $Username
```

```
param (
    [Parameter (Mandatory = $true)]
```

```
[object] $Email,  
[Parameter (Mandatory = $true)]  
[object] $deviceName  
)
```

The screenshot shows the Azure Runbook configuration interface. On the left, there are sections for 'Parameters' with input fields for 'EMAIL' and 'DEVICENAME', 'Run Settings' with a 'Run on' dropdown set to 'Azure', and 'Activity-level tracing' with a 'Trace level' dropdown set to 'None'. On the right, a black box displays the message: 'Click 'Start' to begin the test run. Streams will display when the test completes.'

This is the result view in the Azure Portal if parameters are defined in the code.

## Deploy

To deploy the code you have to publish the code in the editor view.

The screenshot shows the Azure Runbook editor toolbar. The 'Publish' button, represented by a globe icon, is highlighted with a red rectangular box. Other buttons include 'Save', 'Revert to published', 'Test pane', and 'Feedback'.

After that you are able to start the Runbook via the normal Azure Runbook GUI. This will then use the latest published version of the code.

The screenshot shows the Azure Runbook GUI toolbar. The 'Start' button, represented by a play icon, is highlighted with a red rectangular box. Other buttons include 'View', 'Edit', 'Link to schedule', 'Add webhook', 'Delete', 'Export', and 'Refresh'.

## Things to consider

### Hybrid worker

To use a hybrid worker, you have to deploy an Azure VM, or an Azure Arc enabled VM which is always online. This means that the costs are not very low to use on prem scripts via Azure Runbooks.

### Stacking of requests

Azure Runbook stacks the requests. So, if 100 requests are made, the code execution is asynchronous and you have to wait until all jobs (requests) were processed.

---

Revision #6

Created 2022-12-05 16:36:51 UTC

Updated 2026-03-17 16:15:31 UTC by Luca Noah Caprez