

Contents

Azure Automation User Documentation

Overview

[What is Automation?](#)

[FAQ](#)

[What's new?](#)

[Archive for What's new](#)

Quickstarts

[Create Automation account - Azure portal](#)

[Enable managed identities - Azure portal](#)

[Enable Desired State Configuration for a machine](#)

Tutorials

[Create PowerShell runbook using managed identity](#)

[Create a PowerShell Workflow runbook](#)

[Create a Python 3 runbook](#)

Concepts

[Automation account authentication overview](#)

[Runbook execution overview](#)

[Hybrid Runbook Worker overview](#)

[Automation runbook types](#)

[PowerShell DSC](#)

[Automation network configuration details](#)

Security

[Security controls by Azure Policy](#)

[Security baseline](#)

Data security

[Encryption of secure assets](#)

[Management of Azure Automation data](#)

How-to guides

[Automation Account](#)

[Create Automation account - Azure portal](#)

[Create Automation account - Resource Manager template](#)

[Managed identity](#)

[Using system-assigned managed identity](#)

[Using user-assigned managed identity](#)

[Disable system-assigned managed identity](#)

[Remove user-assigned managed identity](#)

[Troubleshoot managed identity](#)

[Run As account](#)

[Create Run As account](#)

[Delete Run As account](#)

[Manage Run As account](#)

[Configure authentication with Amazon Web Services](#)

[Configure authentication with Azure AD](#)

[Manage DNS records used by Automation](#)

[Connect privately to Automation account](#)

[Manage role permissions and security](#)

[Move Automation account to another subscription](#)

[Delete Automation account](#)

[Migrate from Orchestrator to Azure Automation \(Beta\)](#)

[Shared resources](#)

[Manage certificates](#)

[Manage connections](#)

[Manage credentials](#)

[Manage PowerShell modules](#)

[Manage modules in Azure Automation](#)

[Update Azure PowerShell modules](#)

[Manage schedules](#)

[Manage variables](#)

[Manage Python 2 packages](#)

[Manage Python 3 packages](#)

[Troubleshoot shared resources](#)

Process automation

Use existing runbooks and modules

Learn PowerShell Workflow

Manage runbooks

Author and run runbooks

Edit textual runbooks

Edit Graphical runbooks

Create modular runbooks

Configure runbook input parameters

Test a runbook

Start a runbook

Start a runbook from a webhook

Work with the Graphical runbook SDK

Monitor runbooks

Configure runbook output

Handle errors in graphical runbooks

Forward job data to Azure Monitor Logs

Troubleshoot runbooks

Troubleshoot runbook issues

Data to collect when opening a case for Microsoft Azure Automation

Work with a Hybrid Runbook Worker

Deploy Windows Hybrid Runbook Worker

Deploy Linux Hybrid Runbook Worker

Run runbooks on Hybrid Runbook Worker

Use Azure Policy to enforce job execution

Troubleshoot Hybrid Runbook Worker issues

Use source control integration

Configuration Management

Azure Automation State Configuration

Overview

Configure Linux desired state - PowerShell

Get started with State Configuration

[Enable State Configuration](#)

[Configure servers to a desired state and manage drift](#)

[Compose DSC configurations](#)

[Compile DSC configurations](#)

[Remediate noncompliant State Configuration servers](#)

[Remove node and configuration](#)

[Set up continuous deployment with Chocolatey](#)

[Integrate with Azure Monitor Logs](#)

[Work with State Configuration extension version history](#)

[Troubleshoot State Configuration issues](#)

[Scenarios](#)

[Configure data based on STIG](#)

[Configure data at scale](#)

[Create config from existing servers](#)

[Convert configurations to composite resources](#)

[Change tracking and inventory](#)

[Overview](#)

[Support regions for linked Log Analytics workspace](#)

[Enable](#)

[Enable from the Azure portal](#)

[Enable from an Azure VM](#)

[Enable from an Automation account](#)

[Enable from a runbook](#)

[Manage change tracking and inventory](#)

[Manage inventory collection from VMs](#)

[Work with scope configurations](#)

[Configure alerts](#)

[Disable](#)

[Remove Change Tracking and Inventory](#)

[Remove VMs from Change Tracking and Inventory](#)

[Troubleshoot](#)

[Troubleshoot feature deployment issues](#)

Troubleshoot Change Tracking and Inventory issues

Start/Stop VMs during off-hours

Overview

Support regions for linked Log Analytics workspace

Enable Start/Stop VMs during off-hours

Configure Stop/Start VMs during off-hours

Query logs from Start/Stop VMs during off-hours

Remove Start/Stop VMs during off-hours

Troubleshoot Stop/Start VMs during off-hours

Update Management

Overview

Supported regions for linked Log Analytics workspace

Plan your deployment

Enable

System prerequisites

Enable using Azure Resource Manager template

Enable from the Azure portal

Enable from an Azure VM

Enable from an Automation account

Enable from a runbook

Manage updates for your VMs

Get started

Assess compliance

Deploy updates and review status

Use pre-scripts and post-scripts

Configure alerts

Integrate with Configuration Manager

Configure Windows Update client

Use dynamic groups

Query Update Management logs

Work with scope configurations

Disable

- [Remove Update Management](#)
- [Remove VMs from Update Management](#)
- [Troubleshoot](#)
 - [Troubleshoot feature deployment issues](#)
 - [Troubleshoot Update Management issues](#)
 - [Troubleshoot Windows update agent issues](#)
 - [Troubleshoot Linux update agent issues](#)
- [Scenarios](#)
 - [Send an email from a runbook](#)
 - [Monitor runbooks with metric alert](#)
 - [Trigger runbook from Azure alert](#)
 - [Track updated files with watcher task](#)
 - [Manage Office 365 services](#)
 - [Deploy AWS VM with Automation runbook](#)
 - [Deploy Resource Manager template with runbook](#)
 - [Integrate with Event Grid and Microsoft Teams](#)
 - [Automate start/stop of Azure-SSIS IR](#)
- [Reference](#)
 - [Azure CLI](#)
 - [Azure PowerShell Az](#)
 - [Azure PowerShell AzureRM](#)
 - [.NET](#)
 - [REST](#)
 - [Azure Policy built-ins](#)
- [Resources](#)
 - [Automation introduction video](#)
 - [Azure Roadmap](#)
 - [Microsoft Q&A question page](#)
 - [Pricing](#)
 - [Pricing calculator](#)
 - [Release notes](#)
 - [Service updates](#)

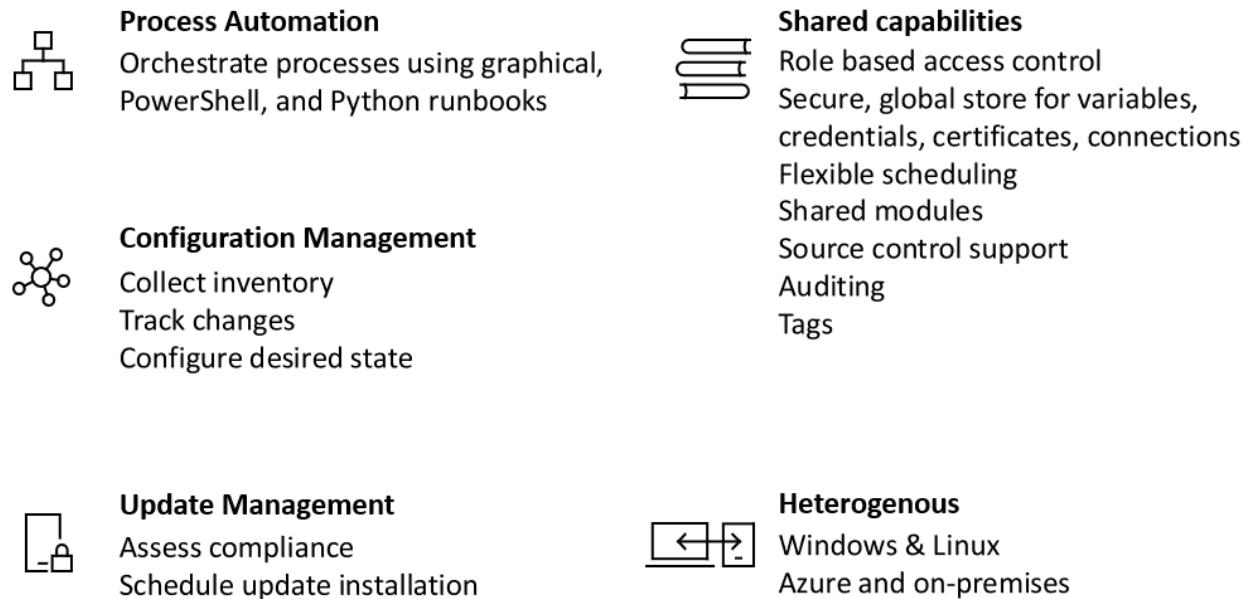
Stack Overflow

Videos

An introduction to Azure Automation

9/10/2021 • 4 minutes to read • [Edit Online](#)

Azure Automation delivers a cloud-based automation and configuration service that supports consistent management across your Azure and non-Azure environments. It comprises process automation, configuration management, update management, shared capabilities, and heterogeneous features. Automation gives you complete control during deployment, operations, and decommissioning of workloads and resources.



Process Automation

Process Automation in Azure Automation allows you to automate frequent, time-consuming, and error-prone cloud management tasks. This service helps you focus on work that adds business value. By reducing errors and boosting efficiency, it also helps to lower your operational costs. The process automation operating environment is detailed in [Runbook execution in Azure Automation](#).

Process automation supports the integration of Azure services and other public systems required in deploying, configuring, and managing your end-to-end processes. The service allows you to author [runbooks](#) graphically, in PowerShell, or using Python. By using a [Hybrid Runbook Worker](#), you can unify management by orchestrating across on-premises environments. [Webhooks](#) let you fulfill requests and ensure continuous delivery and operations by triggering automation from ITSM, DevOps, and monitoring systems.

Configuration Management

Configuration Management in Azure Automation allows access to two features:

- Change Tracking and Inventory
- Azure Automation State Configuration

Change Tracking and Inventory

Change Tracking and Inventory combines change tracking and inventory functions to allow you to track virtual machine and server infrastructure changes. The service supports change tracking across services, daemons, software, registry, and files in your environment to help you diagnose unwanted changes and raise alerts. Inventory support allows you to query in-guest resources for visibility into installed applications and other

configuration items. For details of this feature, see [Change Tracking and Inventory](#).

Azure Automation State Configuration

[Azure Automation State Configuration](#) is a cloud-based feature for PowerShell desired state configuration (DSC) that provides services for enterprise environments. Using this feature, you can manage your DSC resources in Azure Automation and apply configurations to virtual or physical machines from a DSC pull server in the Azure cloud.

Update management

Azure Automation includes the [Update Management](#) feature for Windows and Linux systems across hybrid environments. Update Management gives you visibility into update compliance across Azure and other clouds, and on-premises. The feature allows you to create scheduled deployments that orchestrate the installation of updates within a defined maintenance window. If an update shouldn't be installed on a machine, you can use Update Management functionality to exclude it from a deployment.

Shared capabilities

Azure Automation offers a number of shared capabilities, including shared resources, role-based access control, flexible scheduling, source control integration, auditing, and tagging.

Shared resources

Azure Automation consists of a set of shared resources that make it easier to automate and configure your environments at scale.

- [Schedules](#) - Trigger Automation operations at predefined times.
- [Modules](#) - Manage Azure and other systems. You can import modules into the Automation account for Microsoft, third-party, community, and custom-defined cmdlets and DSC resources.
- [Modules gallery](#) - Supports native integration with the PowerShell Gallery to let you view runbooks and import them into the Automation account. The gallery allows you to quickly get started integrating and authoring your processes from PowerShell gallery and Microsoft Script Center.
- [Python 2 packages](#) - Support Python 2 runbooks for your Automation account.
- [Credentials](#) - Securely store sensitive information that runbooks and configurations can use at runtime.
- [Connections](#) - Store name-value pairs of common information for connections to systems. The module author defines connections in runbooks and configurations for use at runtime.
- [Certificates](#) - Define information to be used in authentication and securing of deployed resources when accessed by runbooks or DSC configurations at runtime.
- [Variables](#) - Hold content that can be used across runbooks and configurations. You can change variable values without having to modify any of the runbooks or configurations that reference them.

Role-based access control

Azure Automation supports Azure role-based access control (Azure RBAC) to regulate access to the Automation account and its resources. To learn more about configuring Azure RBAC on your Automation account, runbooks, and jobs, see [Role-based access control for Azure Automation](#).

Source control integration

Azure Automation supports [source control integration](#). This feature promotes configuration as code where runbooks or configurations can be checked into a source control system.

Heterogeneous support (Windows and Linux)

Automation is designed to work across your hybrid cloud environment and also your Windows and Linux systems. It delivers a consistent way to automate and configure deployed workloads and the operating systems

that run them.

Common scenarios for Automation

Azure Automation supports management throughout the lifecycle of your infrastructure and applications.

Common scenarios include:

- **Write runbooks** - Author PowerShell, PowerShell Workflow, graphical, Python 2, and DSC runbooks in common languages.
- **Build and deploy resources** - Deploy virtual machines across a hybrid environment using runbooks and Azure Resource Manager templates. Integrate into development tools, such as Jenkins and Azure DevOps.
- **Configure VMs** - Assess and configure Windows and Linux machines with configurations for the infrastructure and application.
- **Share knowledge** - Transfer knowledge into the system on how your organization delivers and maintains workloads.
- **Retrieve inventory** - Get a complete inventory of deployed resources for targeting, reporting, and compliance.
- **Find changes** - Identify changes that can cause misconfiguration and improve operational compliance.
- **Monitor** - Isolate machine changes that are causing issues and remediate or escalate them to management systems.
- **Protect** - Quarantine machines if security alerts are raised. Set in-guest requirements.
- **Govern** - Set up Azure RBAC for teams. Recover unused resources.

NOTE

This service supports [Azure Lighthouse](#), which lets service providers sign in to their own tenant to manage subscriptions and resource groups that customers have delegated.

Pricing for Azure Automation

You can review the prices associated with Azure Automation on the [pricing](#) page.

Next steps

[Create an Automation account](#)

Azure Automation frequently asked questions

6/10/2021 • 5 minutes to read • [Edit Online](#)

This Microsoft FAQ is a list of commonly asked questions about Azure Automation. If you have any other questions about its capabilities, go to the discussion forum and post your questions. When a question is frequently asked, we add it to this article so that it's found quickly and easily.

Update Management

Can I prevent unexpected OS-level upgrades?

On some Linux variants, such as Red Hat Enterprise Linux, OS-level upgrades might occur through packages. This might lead to Update Management runs in which the OS version number changes. Because Update Management uses the same methods to update packages that an administrator uses locally on a Linux machine, this behavior is intentional.

To avoid updating the OS version through Update Management deployments, use the **Exclusion** feature.

In Red Hat Enterprise Linux, the package name to exclude is `redhat-release-server.x86_64`.

Why aren't critical/security updates applied?

When you deploy updates to a Linux machine, you can select update classifications. This option filters the updates that meet the specified criteria. This filter is applied locally on the machine when the update is deployed.

Because Update Management performs update enrichment in the cloud, you can flag some updates in Update Management as having a security impact, even though the local machine doesn't have that information. If you apply critical updates to a Linux machine, there might be updates that aren't marked as having a security impact on that machine and therefore aren't applied. However, Update Management might still report that machine as noncompliant because it has additional information about the relevant update.

Deploying updates by update classification doesn't work on RTM versions of CentOS. To properly deploy updates for CentOS, select all classifications to make sure updates are applied. For SUSE, selecting **ONLY Other updates** as the classification can install some other security updates if they're related to zypper (package manager) or its dependencies are required first. This behavior is a limitation of zypper. In some cases, you might be required to rerun the update deployment and then verify the deployment through the update log.

Can I deploy updates across Azure tenants?

If you have machines that need patching in another Azure tenant reporting to Update Management, you must use a following workaround to get them scheduled. You can use the `New-AzAutomationSchedule` cmdlet with the `ForUpdateConfiguration` parameter specified to create a schedule. You can use the `New-AzAutomationSoftwareUpdateConfiguration` cmdlet and pass the machines in the other tenant to the `NonAzureComputer` parameter. The following example shows how to do this.

```

$nonAzurecomputers = @("server-01", "server-02")

$startTime = ([DateTime]::Now).AddMinutes(10)

$sched = New-AzAutomationSchedule -ResourceGroupName mygroup -AutomationAccountName myaccount -Name myupdateconfig -Description test-OneTime -OneTime -StartTime $startTime -ForUpdateConfiguration

New-AzAutomationSoftwareUpdateConfiguration -ResourceGroupName $rg -AutomationAccountName <automationAccountName> -Schedule $sched -Windows -NonAzureComputer $nonAzurecomputers -Duration (New-TimeSpan -Hours 2) -IncludedUpdateClassification Security,UpdateRollup -ExcludedKbNumber KB01,KB02 - IncludedKbNumber KB100

```

Process automation - Python runbooks

Which Python 3 version is supported in Azure Automation?

For cloud jobs, Python 3.8 is supported. Scripts and packages from any 3.x version might work if the code is compatible across different versions.

For hybrid jobs on Windows Hybrid Runbook Workers, you can choose to install any 3.x version you want to use. For hybrid jobs on Linux Hybrid Runbook Workers, we depend on Python 3 version installed on the machine to run DSC OMSConfig and the Linux Hybrid Worker. We recommend installing version 3.6; however, different versions should also work if there are no breaking changes in method signatures or contracts between versions of Python 3.

Can Python 2 and Python 3 runbooks run in same Automation account?

Yes, there's no limitation for using Python 2 and Python 3 runbooks in same Automation account.

What is the plan for migrating existing Python 2 runbooks and packages to Python 3?

Azure Automation doesn't plan to migrate Python 2 runbooks and packages to Python 3. You'll have to do this migration yourself. Existing and new Python 2 runbooks and packages will continue to work.

What are the packages supported by default in Python 3 environment?

Azure package 4.0.0 is installed by default in Python 3 Automation environment. You can manually import a higher version of Azure package to override the default version.

What if I run a Python 3 runbook that references a Python 2 package or the other way around?

Python 2 and Python 3 have different execution environments. While a Python 2 runbook is running, only Python 2 packages can be imported and similar for Python 3.

How do I differentiate between Python 2 and Python 3 runbooks and packages?

Python 3 is a new runbook definition, which distinguishes between Python 2 and Python 3 runbooks. Similarly, another package kind is introduced for Python 3 packages.

How does a Hybrid Runbook Worker know which version of Python to run when both Python2 and Python3 are installed?

For a Windows Runbook Worker, when running a Python 2 runbook it looks for the environment variable `PYTHON_2_PATH` first and validates whether it points to a valid executable file. For example, if the installation folder is `C:\Python2`, it would check if `C:\Python2\python.exe` is a valid path. If not found, then it looks for the `PATH` environment variable to do a similar check.

For Python 3, it looks for the `PYTHON_3_PATH` env variable first and then falls back to the `PATH` environment variable.

When using only one version of Python, you can add the installation path to the `PATH` variable. If you want to use both versions on the Runbook Worker, set `PYTHON_2_PATH` and `PYTHON_3_PATH` to the location of the module for those versions.

How does a Hybrid Runbook Worker locate the Python interpreter?

Locating the Python module is controlled by environment variables as explained earlier.

Is Python 3 supported in Source Control?

No. Source Control isn't currently supported for Python 3. By default, Python runbooks are synced as Python 2 runbooks.

How can a runbook author know what Python packages are available in an Azure sandbox?

Use the following code to list the default installed modules:

```
#!/usr/bin/env python3

import pkg_resources
installed_packages = pkg_resources.working_set
installed_packages_list = sorted(["%s==%s" % (i.key, i.version)
    for i in installed_packages])

for package in installed_packages_list:
    print(package)
```

How can a runbook author set which version of a package module to be used if there are multiple modules?

The default version can be overridden by importing the Python packages in the Automation account. Preference is given to the imported version in the Automation account.

Next steps

If your question isn't answered here, you can refer to the following sources for more questions and answers.

- [Azure Automation](#)
- [Feedback forum](#)

What's new in Azure Automation?

8/27/2021 • 5 minutes to read • [Edit Online](#)

Azure Automation receives improvements on an ongoing basis. To stay up to date with the most recent developments, this article provides you with information about:

- The latest releases
- Known issues
- Bug fixes

This page is updated monthly, so revisit it regularly. If you're looking for items older than six months, you can find them in [Archive for What's new in Azure Automation](#).

August 2021

Azure Policy Guest Configuration

Type: Plan for change

Customers should evaluate and plan for migration from Azure Automation State Configuration to Azure Policy guest configuration. For more information, see [Azure Policy guest configuration](#).

July 2021

Preview support for user-assigned managed identity

Type: New feature

Azure Automation now supports [user-assigned Managed Identities](#) for cloud jobs in Azure global, Azure Government, and Azure China regions. Read the [announcement](#) for more information.

General Availability of customer-managed keys for Azure Automation

Type: New feature

Customers can manage and secure encryption of Azure Automation assets using their own managed keys. With the introduction of customer-managed keys you can supplement default encryption with an additional encryption layer using keys that you create and manage in Azure Key Vault. This additional encryption should help you meet your organization's regulatory or compliance needs.

For more information, see [Use of customer-managed keys](#).

June 2021

Security update for Log Analytics Contributor role

Type: Plan for change

Microsoft intends to remove the Automation account rights from the Log Analytics Contributor role. Currently, the built-in [Log Analytics Contributor](#) role can escalate privileges to the subscription [Contributor](#) role. Since Automation account Run As accounts are initially configured with Contributor rights on the subscription, it can be used by an attacker to create new runbooks and execute code as a Contributor on the subscription.

As a result of this security risk, we recommend you don't use the Log Analytics Contributor role to execute Automation jobs. Instead, create the Azure Automation Contributor custom role and use it for actions related to the Automation account. For implementation steps, see [Custom Azure Automation Contributor role](#).

Support for Automation and State Configuration available in West US 3

Type: New feature

For more information, see [Data residency in Azure](#) and select your geography from the drop-down list.

May 2021

Start/Stop VMs during off-hours (v1)

Type: Plan for change

Start/Stop VMs during off-hours (v1) will deprecate on May 21, 2022. Customers should evaluate and plan for migration to the Start/Stop VMs v2 (preview). For more information, see [Start/Stop v2 overview \(preview\)](#).

April 2021

Support for Update Management and Change Tracking

Type: New feature

Region mapping have been updated to support Update Management and Change Tracking in Norway East, UAE North, North Central US, Brazil South, and Korea Central. For more information, see [Supported mappings](#).

Support for system-assigned Managed Identities

Type: New feature

Azure Automation now supports [system-assigned Managed Identities](#) for cloud and Hybrid jobs in Azure global and Azure Government regions. Read the [announcement](#) for more information.

March 2021

New Azure Automation built-in policy definitions

Type: New feature

Azure Automation has added five new built-in policy definitions:

- Automation accounts should disable public network access,
- Azure Automation accounts should use customer-managed keys to encrypt data at rest
- Configure Azure Automation accounts to disable public network access
- Configure private endpoint connections on Azure Automation accounts
- Private endpoint connections on Automation Accounts should be enabled.

For more information, see [Azure Policy reference](#).

Support for Automation and State Configuration declared GA in South India

Type: New feature

Use Process Automation and State Configuration feature in South India. Read the [announcement](#) for more information.

Support for Automation and State Configuration declared GA in UK West

Type: New feature

Use Process Automation and State Configuration feature in UK West. For more information, read [announcement](#).

Support for Automation and State Configuration declared GA in UAE Central

Type: New feature

Use Process Automation and State Configuration feature in UAE Central. Read the [announcement](#) for more information.

Support for Automation and State Configuration available in Australia Central 2, Norway West, and France South

Type: New feature

See more information on the [Data residency page](#) by selecting the geography for each region.

New scripts added for installing Hybrid worker on Windows and Linux

Type: New feature

Two new scripts have been added to the Azure Automation [GitHub repository](#) addressing one of Azure Automation's key scenarios of setting up a Hybrid Runbook Worker on either a Windows or a Linux machine. The script creates a new VM or uses an existing one, creates a Log Analytics workspace if needed, installs the Log Analytics agent for Windows or Log Analytics agent for Linux, and registers the machine to the Log Analytics workspace. The Windows script is named **Create Automation Windows HybridWorker** and the Linux script is **Create Automation Linux HybridWorker**.

Invoke runbook through an Azure Resource Manager template webhook

Type: New feature

For more information, see [Use a webhook from an ARM template](#).

Azure Update Management now supports Centos 8.x, Red Hat Enterprise Linux Server 8.x, and SUSE Linux Enterprise Server 15

Type: New feature

See the [full list](#) of supported Linux operating systems for more details.

In-region data residency support for Brazil South and South East Asia

Type: New feature

In all regions except Brazil South and Southeast Asia, Azure Automation data is stored in a different region (Azure paired region) for providing Business Continuity and Disaster Recovery (BCDR). For the Brazil and Southeast Asia regions only, we now store Azure Automation data in the same region to accommodate data-residency requirements for these regions. For more information, see [Geo-replication in Azure Automation](#).

February 2021

Support for Automation and State Configuration declared GA in Japan West

Type: New feature

Automation account and State Configuration availability in Japan West region. For more information, read [announcement](#).

Introduced custom Azure Policy compliance to enforce runbook execution on Hybrid Worker

Type : New feature

You can use the new Azure Policy compliance rule to allow creation of jobs, webhooks, and job schedules to run only on Hybrid Worker groups.

Update Management availability in East US, France Central, and North Europe regions

Type: New feature

Automation Update Management feature is available in East US, France Central, and North Europe regions. See [Supported region mapping](#) for updates to the documentation reflecting this change.

Next steps

If you'd like to contribute to Azure Automation documentation, see the [Docs Contributor Guide](#).

Archive for What's new in Azure Automation?

8/27/2021 • 6 minutes to read • [Edit Online](#)

The primary [What's new in Azure Automation?](#) article contains updates for the last six months, while this article contains all the older information.

What's new in Azure Automation? provides you with information about:

- The latest releases
- Known issues
- Bug fixes

January 2021

Support for Automation and State Configuration declared GA in Switzerland West

Type: New feature

Automation account and State Configuration availability in the Switzerland West region. For more information, read the [announcement](#).

Added Python 3 script to import module with multiple dependencies

Type: New feature

The script is available for download from our [GitHub repository](#).

Hybrid Runbook Worker role support for Centos 8.x/RHEL 8.x/SLES 15

Type: New feature

The Hybrid Runbook Worker feature supports CentOS 8.x, REHL 8.x, and SLES 15 distributions for only process automation on Hybrid Runbook Workers. See [Supported operating systems](#) for updates to the documentation to reflect these changes.

Update Management and Change Tracking availability in Australia East, East Asia, West US, and Central US regions

Type: New feature

Automation account, Change Tracking and Inventory, and Update Management are available in Australia East, East Asia, West US, and Central US regions.

Introduced public preview of Python 3 runbooks in US Government cloud

Type: New feature Azure Automation introduces public preview support of Python 3 cloud and hybrid runbook execution in US Government cloud regions. For more information, see the [announcement](#).

Azure Automation runbooks moved from TechNet Script Center to GitHub

Type: Plan for change

The TechNet Script Center is retiring and all runbooks hosted in the Runbook gallery have been moved to our [Automation GitHub organization](#). For more information, read [Azure Automation Runbooks moving to GitHub](#).

December 2020

Azure Automation and Update Management Private Link GA

Type: New feature

Azure Automation and Update Management support announced as GA for Azure global and Government clouds. Azure Automation enabled Private Link support to secure execution of a runbook on a hybrid worker role, using Update Management to patch machines, invoking a runbook through a webhook, and using State Configuration service to keep your machines complaint. For more information, read [Azure Automation Private Link support](#)

Azure Automation classified as Grade-C certified on Accessibility

Type: New feature

Accessibility features of Microsoft products help agencies address global accessibility requirements. On the [blog announcement](#) page, search for **Azure Automation** to read the Accessibility conformance report for the Automation service.

Support for Automation and State Configuration GA in UAE North

Type: New feature

Automation account and State Configuration availability in the UAE North region. For more information, read [announcement](#).

Support for Automation and State Configuration GA in Germany West Central

Type: New feature

Automation account and State Configuration availability in Germany West region. For more information, read [announcement](#).

DSC support for Oracle 6 and 7

Type: New feature

Manage Oracle Linux 6 and 7 machines with Automation State Configuration. See [Supported Linux distros](#) for updates to the documentation to reflect these changes.

Public Preview for Python3 runbooks in Automation

Type: New feature

Azure Automation now supports Python 3 cloud and hybrid runbook execution in public preview in all regions in Azure global cloud. For more information, see the [announcement] (<https://azure.microsoft.com/updates/azure-automation-python-3-public-preview/>).

November 2020

DSC support for Ubuntu 18.04

Type: New feature

See [Supported Linux Distros](#) for updates to the documentation reflecting these changes.

October 2020

Support for Automation and State Configuration GA in Switzerland North

Type: New feature

Automation account and State Configuration availability in Switzerland North. For more information, read [announcement](#).

Support for Automation and State Configuration GA in Brazil South East

Type: New feature

Automation account and State Configuration availability in Brazil South East. For more information, read [announcement](#).

Update Management availability in South Central US

Type: New feature

Azure Automation region mapping updated to support Update Management feature in South Central US region. See [Supported region mapping](#) for updates to the documentation to reflect this change.

September 2020

Start/Stop VMs during off-hours runbooks updated to use Azure Az modules

Type: New feature

Start/Stop VM runbooks have been updated to use Az modules in place of Azure Resource Manager modules.

See [Start/Stop VMs during off-hours](#) overview for updates to the documentation to reflect these changes.

August 2020

Published the DSC extension to support Azure Arc

Type: New feature

Use Azure Automation State Configuration to centrally store configurations and maintain the desired state of hybrid connected machines enabled through the Azure Arc enabled servers DSC VM extension. For more information, read [Arc enabled servers VM extensions overview](#).

July 2020

Introduced Public Preview of Private Link support in Automation

Type: New feature

Use Azure Private Link to securely connect virtual networks to Azure Automation using private endpoints. For more information, read the [announcement](#).

Hybrid Runbook Worker support for Windows Server 2008 R2

Type: New feature

Automation Hybrid Runbook Worker supports the Windows Server 2008 R2 operating system. See [Supported operating systems](#) for updates to the documentation to reflect these changes.

Update Management support for Windows Server 2008 R2

Type: New feature

Update Management supports assessing and patching the Windows Server 2008 R2 operating system. See [Supported operating systems](#) for updates to the documentation to reflect these changes.

Automation diagnostic logs schema update

Type: New feature

Changed the schema of Azure Automation log data in the Log Analytics service. To learn more, see [Forward Azure Automation job data to Azure Monitor logs](#).

Azure Lighthouse supports Automation Update Management

Type: New feature

Azure Lighthouse enables delegated resource management with Update Management for service providers and customers. Read more [here](#).

June 2020

Automation and Update Management availability in the US Gov Arizona region

Type: New feature

Automation account and Update Management are available in US Gov Arizona. For more information, see [announcement](#).

Hybrid Runbook Worker onboarding script updated to use Az modules

Type: New feature

The New-OnPremiseHybridWorker runbook has been updated to support Az modules. For more information, see the package in the [PowerShell Gallery](#).

Update Management availability in China East 2

Type: New feature

Azure Automation region mapping updated to support Update Management feature in China East 2 region. See [Supported region mapping](#) for updates to the documentation to reflect this change.

May 2020

Updated Automation service DNS records from region-specific to Automation account-specific URLs

Type: New feature

Azure Automation DNS records have been updated to support Private Links. For more information, read the [announcement](#).

Added capability to keep Automation runbooks and DSC scripts encrypted by default

Type: New feature

In addition to improve security of assets, runbooks, and DSC scripts are also encrypted to enhance Azure Automation security.

April 2020

Retirement of the Automation watcher task

Type: Plan for change

Azure Logic Apps is now the recommended and supported way to monitor for events, schedule recurring tasks, and trigger actions. There will be no further investments in Watcher task functionality. To learn more, see [Schedule and run recurring automated tasks with Logic Apps](#).

March 2020

Support for Impact Level 5 (IL5) compute isolation in Azure commercial and Government cloud

Type:

Azure Automation Hybrid Runbook Worker can be used in Azure Government to support Impact Level 5 workloads. To learn more, see our [documentation](#).

February 2020

Introduced support for Azure virtual network service tags

Type: New feature

Automation support of service tags allows or denies the traffic for the Automation service, for a subset of scenarios. To learn more, see the [documentation](#).

Enable TLS 1.2 support for Azure Automation service

Type: Plan for change

Azure Automation fully supports TLS 1.2 and all client calls (through webhooks, DSC nodes, and hybrid worker). TLS 1.1 and TLS 1.0 are still supported for backward compatibility with older clients until customers standardize and fully migrate to TLS 1.2.

January 2020

Introduced Public Preview of customer-managed keys for Azure Automation

Type: New feature

Customers can manage and secure encryption of Azure Automation assets using their own managed keys. For more information, see [Use of customer-managed keys](#).

Retirement of Azure Service Management (ASM) REST APIs for Azure Automation

Type: Retire

Azure Service Management (ASM) REST APIs for Azure Automation will be retired and no longer supported after January 30, 2020. To learn more, see the [announcement](#).

Next steps

If you'd like to contribute to Azure Automation documentation, see the [Docs Contributor Guide](#).

Quickstart: Create an Automation account using the Azure portal

9/5/2021 • 2 minutes to read • [Edit Online](#)

You can create an Azure [Automation account](#) using the Azure portal, a browser-based user interface allowing access to a number of resources. One Automation account can manage resources across all regions and subscriptions for a given tenant. This Quickstart guides you in creating an Automation account.

Prerequisites

An Azure account with an active subscription. [Create an account for free](#).

Create Automation account

1. Sign in to the [Azure portal](#).
2. From the top menu, select + **Create a resource**.
3. Under Categories**, select **IT & Management Tools**, and then select **Automation**.

The screenshot shows the Microsoft Azure portal's 'Create a resource' interface. On the left, a sidebar lists categories under 'IT & Management Tools'. The 'Automation' item is highlighted with a red box. The main area displays a list of popular products from the marketplace, each with a 'Create' button. The 'Automation' product is also highlighted with a red box.

Category	Product	Action
IT & Management Tools	Waterline AI-Driven Data Catalog	Create Learn more
	VeriChannel as a Service	Create Learn more
	Nerdio Manager for Enterprise	Create Learn more
	Automation	Create Learn more
	DxEnterprise for Availability Groups (AGs)	Create Learn more
	Qorus Integration Engine® 4.1.x on Oracle Linux 7	Create Learn more
	SIMBA Chain Smart Contract as a Service	Create Learn more
	Media	
	Migration	
	Mixed Reality	

4. From the Add Automation Account page, provide the following information:

PROPERTY	DESCRIPTION
Name	Enter a name unique for its location and resource group. Names for Automation accounts that have been deleted might not be immediately available. You can't change the account name once it has been entered in the user interface.
Subscription	From the drop-down list, select the Azure subscription for the account.
Resource group	From the drop-down list, select your existing resource group, or select Create new .
Location	From the drop-down list, select a location for the account. For an updated list of locations that you can deploy an Automation account to, see Products available by region
Create Azure Run As account	Select No . An Azure Run As account in the Automation account is useful for authenticating with Azure; however, managed identities in Automation is now available. Managed identities provide an identity for applications to use when connecting to resources that support Azure Active Directory (Azure AD) authentication.

Add Automation Account ...

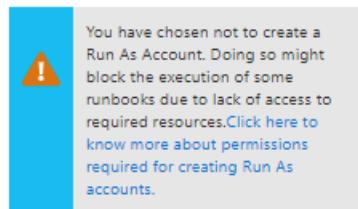
Name * ⓘ
 ✓

Subscription *
 ▼

Resource group *
 ▼
[Create new](#)

Location *
 ▼

Create Azure Run As account * ⓘ
Yes No



Create

5. Select **Create** to start the Automation account deployment. The creation completes in about a minute.
6. You will receive a notification when the deployment has completed. Select **Go to resource** in the notification to open the **Automation Account** page.
7. Review your new Automation account.

myAutomationAccount

Automation Account

Search (Ctrl+ /)

Delete Move Feedback Refresh

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Configuration Management

Inventory

Change tracking

State configuration (DSC)

Update management

Update management

Process Automation

Runbooks

Jobs

Runbooks gallery

Hybrid worker groups

Watcher tasks

Shared Resources

Schedules

Modules

If you've lost access to Automation Account or getting unauthorized for any operations, this could be related to the change made to the Log Analytics Contributor Role. Follow <https://aka.ms/custom-automation-contributor-role> & <https://aka.ms/automation-contributor-update-management> to mitigate the problem.

Essentials

Resource group (change) contosoGroup

Subscription ID Contoso ID

Location West US

Status Active

Last modified 9/3/2021, 9:21 AM

Tags (change)
Click here to add tags

Job Statistics
Last 24 Hours

0

Failed 0

Suspended 0

Completed 0

Running 0

Queued 0

Stopped 0

JSON View

The screenshot shows the Azure portal's 'Overview' page for an automation account named 'myAutomationAccount'. The main content area includes a summary of the account's configuration (Resource group: contosoGroup, Subscription ID: Contoso ID, Location: West US, Status: Active), a section for adding tags, and a large circular gauge showing '0' for job statistics over the last 24 hours across six categories: Failed, Suspended, Completed, Running, Queued, and Stopped. On the left, a sidebar provides navigation links for activity log, access control, tags, diagnosis, configuration management, inventory, change tracking, state configuration, update management, process automation (runbooks, jobs, runbooks gallery, hybrid worker groups, watcher tasks), shared resources (schedules, modules), and a general search bar at the top.

Clean up resources

If you're not going to continue to use the Automation account, select **Delete** from the **Overview** page, and then select **Yes** when prompted.

Next steps

In this Quickstart, you created an Automation account. To use managed identities with your Automation account, continue to the next Quickstart:

[Quickstart - Enable managed identities](#)

Quickstart: Enable managed identities for your Automation account using the Azure portal

9/5/2021 • 2 minutes to read • [Edit Online](#)

This Quickstart shows you how to enable managed identities for an Azure Automation account. For more information on how managed identities work with Azure Automation, see [Managed identities](#).

Prerequisites

- An Azure account with an active subscription. [Create an account for free](#).
- An Azure Automation account. For instructions, see [Create an Automation account](#).
- A user-assigned managed identity. For instructions, see [Create a user-assigned managed identity](#). The user-assigned managed identity and the target Azure resources that your runbook manages using that identity must be in the same Azure subscription.

Enable system-assigned managed identity

1. Sign in to the [Azure portal](#) and navigate to your Automation account.
2. Under **Account Settings**, select **Identity (Preview)**.

The screenshot shows the Azure portal interface for managing an Automation account named 'myAutomationAccount'. The left sidebar lists various settings like Certificates, Variables, and Run as accounts, with 'Identity (Preview)' highlighted. The main content area shows the 'Identity (Preview)' settings. It includes tabs for 'System assigned' (which is selected and highlighted with a red box) and 'User assigned'. A descriptive text explains that a system-assigned managed identity is restricted to one per resource and is tied to the lifecycle of the resource, using Azure RBAC for authentication. Below this is a 'Save' button (also highlighted with a red box), a 'Discard' button, a 'Refresh' button, and a 'Got feedback?' link. At the bottom, there is a 'Status' switch with options 'Off' and 'On' (which is selected and highlighted with a red box).

3. Set the system-assigned **Status** option to **On** and then press **Save**. When you're prompted to confirm, select **Yes**.

Your Automation account can now use the system-assigned identity, which is registered with Azure Active Directory (Azure AD) and is represented by an object ID.

Status: On

Object (principal) ID: bde1d75a-f6a4-42a5-a5ae-015ba61176f7

Permissions: Azure role assignments

This resource is registered with Azure Active Directory. The managed identity can be configured to allow access to other resources. Be careful when making changes to the access settings for the managed identity because it can result in failures. [Learn more](#)

Add user-assigned managed identity

This section continues from where the last section ended.

1. Select the **User assigned** tab, and then select **+ Add** or **Add user assigned managed identity** to open the **Add user assigned managed i...** page.

System assigned User assigned

User assigned managed identities enable Azure resources to authenticate to cloud services (e.g. Azure Key Vault) without storing credentials in code. This type of managed identities are created as standalone Azure resources, and have their own lifecycle. A single resource (e.g. Virtual Machine) can utilize multiple user assigned managed identities. Similarly, a single user assigned managed identity can be shared across multiple resources (e.g. Virtual Machine). [Learn more about Managed identities.](#)

+ Add Remove Refresh Got feedback?

Name	resource group	subscription
No results		

No user assigned managed identities found on this resource

Add user assigned managed identity

2. From the **Subscription** drop-down list, select the subscription for your user-assigned managed identity.

Add user assigned managed i... X

Subscription *

Contoso

User assigned managed identities

Filter by identity name and/or resource group name

myFirstID
Resource Group: groupContoso

Selected identities:

myFirstID groupContoso Remove

Add

3. Under **User assigned managed identities**, select your existing user-assigned managed identity and then select **Add**. You'll then be returned to the **User assigned** tab.

System assigned User assigned

User assigned managed identities enable Azure resources to authenticate to cloud services (e.g. Azure Key Vault) without storing credentials in code. This type of managed identities are created as standalone Azure resources, and have their own lifecycle. A single resource (e.g. Virtual Machine) can utilize multiple user assigned managed identities. Similarly, a single user assigned managed identity can be shared across multiple resources (e.g. Virtual Machine). [Learn more about Managed identities.](#)

+ Add Remove Refresh | Got feedback?

Name	resource group	subscription
myfirstid	contosoGroup	Contoso ID

Clean up resources

If you no longer need the user-assigned managed identity attached to your Automation account, perform the following steps:

1. From the **User assigned** tab, select your user-assigned managed identity.
2. From the top menu, select **Remove**, and then select **Yes** when prompted for confirmation.

If you no longer need the system-assigned managed identity enabled for your Automation account, perform the following steps:

1. From the **System assigned** tab, under **Status**, select **Off**.
2. From the top menu, select **Save**, and then select **Yes** when prompted for confirmation.

Next steps

In this Quickstart, you enabled managed identities for an Azure Automation account. To use your Automation account with managed identities to execute a runbook, see.

[Tutorial: Create Automation PowerShell runbook using managed identity](#)

Configure a VM with Desired State Configuration

9/1/2021 • 4 minutes to read • [Edit Online](#)

By enabling Azure Automation State Configuration, you can manage and monitor the configurations of your Windows and Linux servers using Desired State Configuration (DSC). Configurations that drift from a desired configuration can be identified or auto-corrected. This quickstart steps through enabling an Azure Linux VM and deploying a LAMP stack using Azure Automation State Configuration.

Prerequisites

To complete this quickstart, you need:

- An Azure subscription. If you don't have an Azure subscription, [create a free account](#).
- An Azure Automation account. For instructions on creating an Azure Automation Run As account, see [Azure Run As Account](#).
- An Azure Resource Manager virtual machine running Red Hat Enterprise Linux, CentOS, or Oracle Linux. For instructions on creating a VM, see [Create your first Linux virtual machine in the Azure portal](#)

Sign in to Azure

Sign in to Azure at <https://portal.azure.com>.

Enable a virtual machine

There are many different methods to enable a machine for Automation State Configuration. This quickstart tells how to enable the feature for an Azure VM using an Automation account. You can learn more about different methods to enable your machines for State Configuration by reading [Enable machines for management by Azure Automation State Configuration](#).

1. In the Azure portal, navigate to **Automation accounts**.
2. From the list of Automation accounts, select an account.
3. From the left pane of the Automation account, select **State configuration (DSC)**.
4. Click **Add** to open the **VM select** page.
5. Find the virtual machine for which to enable DSC. You can use the search field and filter options to find a specific virtual machine.
6. Click on the virtual machine, and then click **Connect**
7. Select the DSC settings appropriate for the virtual machine. If you have already prepared a configuration, you can specify it as **Node Configuration Name**. You can set the **configuration mode** to control the configuration behavior for the machine.
8. Click **OK**. While the DSC extension is deployed to the virtual machine, the status reported is **Connecting**.

Import modules

Modules contain DSC resources and many can be found in the [PowerShell Gallery](#). Any resources that are used in your configurations must be imported to the Automation account before compiling. For this quickstart, the module named **nx** is required.

1. From the left pane of the Automation account, select **Modules Gallery** under Shared Resources.
2. Search for the module to import by typing part of its name: `nx`.
3. Click on the module to import.
4. Click **Import**.

Module Name	Description	Created by	Downloads	Last updated
nx	Module with DSC Resources for Linux	MSFT_OSTC	3008	9/25/2015
nxNetworking	Module with DSC Networking Resources for Linux	MSFT_OSTC	803	9/29/2015
nxComputerManagement	Module with DSC Computer Management Resources for Linux	MSFT_OSTC	769	9/29/2015
cEPRSRUnXVT	DSC script for executing X-Verification Tests(EVT, IVT & BVT)	EPRSBuild	204	9/29/2015

Import the configuration

This quickstart uses a DSC configuration that configures Apache HTTP Server, MySQL, and PHP on the machine. See [DSC configurations](#).

In a text editor, type the following and save it locally as **AMPServer.ps1**.

```

configuration 'LAMPServer' {
    Import-DSCResource -module "nx"

    Node localhost {

        $requiredPackages = @("httpd","mod_ssl","php","php-mysql","mariadb","mariadb-server")
        $enabledServices = @("httpd","mariadb")

        #Ensure packages are installed
        ForEach ($package in $requiredPackages){
            nxPackage $Package{
                Ensure = "Present"
                Name = $Package
                PackageManager = "yum"
            }
        }

        #Ensure daemons are enabled
        ForEach ($service in $enabledServices){
            nxService $service{
                Enabled = $true
                Name = $service
                Controller = "SystemD"
                State = "running"
            }
        }
    }
}

```

To import the configuration:

1. In the left pane of the Automation account, select **State configuration (DSC)** and then click the **Configurations** tab.
2. Click **+ Add**.
3. Select the configuration file that you saved in the prior step.
4. Click **OK**.

Compile a configuration

You must compile a DSC configuration to a node configuration (MOF document) before it can be assigned to a node. Compilation validates the configuration and allows for the input of parameter values. To learn more about compiling a configuration, see [Compiling configurations in State Configuration](#).

1. In the left pane of the Automation account, select **State Configuration (DSC)** and then click the **Configurations** tab.
2. Select the configuration **LAMPServer**.
3. From the menu options, select **Compile** and then click **Yes**.
4. In the Configuration view, you see a new compilation job queued. When the job has completed successfully, you are ready to move on to the next step. If there are any failures, you can click on the compilation job for details.

Assign a node configuration

You can assign a compiled node configuration to a DSC node. Assignment applies the configuration to the machine and monitors or auto-corrects for any drift from that configuration.

1. In the left pane of the Automation account, select **State Configuration (DSC)** and then click the **Nodes** tab.
2. Select the node to which to assign a configuration.

3. Click **Assign Node Configuration**
4. Select the node configuration `LAMPServer.localhost` and click **OK**. State Configuration now assigns the compiled configuration to the node, and the node status changes to `Pending`. On the next periodic check, the node retrieves the configuration, applies it, and reports status. It can take up to 30 minutes for the node to retrieve the configuration, depending on the node settings.
5. To force an immediate check, you can run the following command locally on the Linux virtual machine:

```
sudo /opt/microsoft/dsc/Scripts/PerformRequiredConfigurationChecks.py
```

LinuxVM2

Assign node configuration **Unregister**

Essentials

Resource group	IP address
ExampleAutoAccount	127.0.0.1
Id	Account
e966fcbc-96a2-484a-9c8c-afa66661f45b	ExampleAutoAccount
Last seen time	Virtual machine
11/6/2018, 12:17 PM	LinuxVM2
Configuration	Node configuration
--	--
Registration time	Status
11/6/2018, 12:17 PM	Compliant

Reports

TYPE	STATUS	REPORT TIME
Consistency	✓ Compliant	11/6/2018, 12:17 PM

View node status

You can view the status of all State Configuration-managed nodes in your Automation account. The information is displayed by choosing **State Configuration (DSC)** and clicking the **Nodes** tab. You can filter the display by status, node configuration, or name search.

ExampleAutoAccount - State configuration (DSC)

Nodes

NODE	STATUS	NODE CONFIGURATION	LAST SEEN	VERSION
LinuxVM2	Pending	LAMPServer.localhost	11/6/2018, 12:30 PM	2.70.0.10
LinuxVM3	Compliant	LAMPServer.localhost	11/6/2018, 12:30 PM	2.70.0.10

Next steps

In this quickstart, you enabled an Azure Linux VM for State Configuration, created a configuration for a LAMP stack, and deployed the configuration to the VM. To learn how you can use Azure Automation State Configuration to enable continuous deployment, continue to the article:

[Set up continuous deployment with Chocolatey](#)

Tutorial: Create Automation PowerShell runbook using managed identity

9/5/2021 • 4 minutes to read • [Edit Online](#)

This tutorial walks you through creating a [PowerShell runbook](#) in Azure Automation that uses [managed identities](#), rather than the Run As account to interact with resources. PowerShell runbooks are based on Windows PowerShell. A managed identity from Azure Active Directory (Azure AD) allows your runbook to easily access other Azure AD-protected resources.

In this tutorial, you learn how to:

- Assign permissions to managed identities
- Create a PowerShell runbook

If you don't have an Azure subscription, create a [free account](#) before you begin.

Prerequisites

- An Azure Automation account with at least one user-assigned managed identity. For more information, see [Using a user-assigned managed identity for an Azure Automation account](#).
- Az modules: `Az.Accounts`, `Az.Automation`, `Az.ManagedServiceIdentity`, and `Az.Compute` imported into the Automation account. For more information, see [Import Az modules](#).
- The [Azure Az PowerShell module](#) installed on your machine. To install or upgrade, see [How to install the Azure Az PowerShell module](#).
- An [Azure virtual machine](#). Since you stop and start this machine, it shouldn't be a production VM.
- A general familiarity with [Automation runbooks](#).

Assign permissions to managed identities

Assign permissions to the managed identities to allow them to stop and start a virtual machine.

1. Sign in to Azure interactively using the [Connect-AzAccount](#) cmdlet and follow the instructions.

```
# Sign in to your Azure subscription
$sub = Get-AzSubscription -ErrorAction SilentlyContinue
if(-not($sub))
{
    Connect-AzAccount -Subscription
}

# If you have multiple subscriptions, set the one to use
# Select-AzSubscription -SubscriptionId <SUBSCRIPTIONID>
```

2. Provide an appropriate value for the variables below and then execute the script.

```
$resourceGroup = "resourceGroupName"

# These values are used in this tutorial
$automationAccount = "xAutomationAccount"
$userAssignedOne = "xUAMI"
```

3. Use PowerShell cmdlet [New-AzRoleAssignment](#) to assign a role to the system-assigned managed identity.

```
$role1 = "DevTest Labs User"

$SAMI = (Get-AzAutomationAccount -ResourceGroupName $resourceGroup -Name
$automationAccount).Identity.PrincipalId
New-AzRoleAssignment `

-ObjectId $SAMI `

-ResourceGroupName $resourceGroup `

-RoleDefinitionName $role1
```

4. The same role assignment is needed for the user-assigned managed identity

```
$UAMI = (Get-AzUserAssignedIdentity -ResourceGroupName $resourceGroup -Name
$userAssignedOne).PrincipalId
New-AzRoleAssignment `

-ObjectId $UAMI `

-ResourceGroupName $resourceGroup `

-RoleDefinitionName $role1
```

5. Additional permissions for the system-assigned managed identity are needed to execute cmdlets

`Get-AzUserAssignedIdentity` and `Get-AzAutomationAccount` as used in this tutorial.

```
$role2 = "Reader"
New-AzRoleAssignment `

-ObjectId $SAMI `

-ResourceGroupName $resourceGroup `

-RoleDefinitionName $role2
```

Create PowerShell runbook

Create a runbook that will allow execution by either managed identity. The runbook will start a stopped VM, or stop a running VM.

1. Sign in to the [Azure portal](#), and navigate to your Automation account.
2. Under **Process Automation**, select **Runbooks**.
3. Select **Create a runbook**.
 - a. Name the runbook `miTesting`.
 - b. From the **Runbook type** drop-down menu, select **PowerShell**.
 - c. Select **Create**.
4. In the runbook editor, paste the following code:

```
Param(
    [string]$resourceGroup,
    [string]$VMName,
    [string]$method,
    [string]$UAMI
)

$automationAccount = "xAutomationAccount"

# Ensures you do not inherit an AzContext in your runbook
Disable-AzContextAutosave -Scope Process | Out-Null

# Connect using a Managed Service Identity
```

```

# Connect using a managed service identity
try {
    Connect-AzAccount -Identity -ErrorAction stop -WarningAction SilentlyContinue | Out-Null
}
catch{
    Write-Output "There is no system-assigned user identity. Aborting.";
    exit
}

if ($method -eq "SA")
{
    Write-Output "Using system-assigned managed identity"
}
elseif ($method -eq "UA")
{
    Write-Output "Using user-assigned managed identity"

    # Connects using the Managed Service Identity of the named user-assigned managed identity
    $identity = Get-AzUserAssignedIdentity -ResourceGroupName $resourceGroup -Name $UAMI

    # validates assignment only, not perms
    if ((Get-AzAutomationAccount -ResourceGroupName $resourceGroup -Name
$automationAccount).Identity.UserAssignedIdentities.Values.PrincipalId.Contains($identity.PrincipalId
))
    {
        Connect-AzAccount -Identity -AccountId $identity.ClientId | Out-Null
    }
    else {
        Write-Output "Invalid or unassigned user-assigned managed identity"
        exit
    }
}
else {
    Write-Output "Invalid method. Choose UA or SA."
    exit
}

# Get current state of VM
$status = (Get-AzVM -ResourceGroupName $resourceGroup -Name $VMName -Status).Statuses[1].Code

Write-Output "`r`n Beginning VM status: $status `r`n"

# Start or stop VM based on current state
if($status -eq "Powerstate/deallocated")
{
    Start-AzVM -Name $VMName -ResourceGroupName $resourceGroup
}
elseif ($status -eq "Powerstate/running")
{
    Stop-AzVM -Name $VMName -ResourceGroupName $resourceGroup -Force
}

# Get new state of VM
$status = (Get-AzVM -ResourceGroupName $resourceGroup -Name $VMName -Status).Statuses[1].Code

Write-Output "`r`n Ending VM status: $status `r`n `r`n"

Write-Output "Account ID of current context: " (Get-AzContext).Account.Id

```

5. In the editor, on line 8, revise the value for the `$automationAccount` variable as needed.
6. Select **Save** and then **Test pane**.
7. Populate the parameters `RESOURCEGROUP` and `VMNAME` with the appropriate values. Enter `SA` for the `METHOD` parameter and `xUAMI` for the `UAMI` parameter. The runbook will attempt to change the power state of your VM using the system-assigned managed identity.

8. Select **Start**. Once the runbook completes, the output should look similar to the following:

```
Beginning VM status: PowerState/deallocated

OperationId : 5b707401-f415-4268-9b43-be1f73ddc54b
Status      : Succeeded
StartTime   : 8/3/2021 10:52:09 PM
EndTime     : 8/3/2021 10:52:50 PM
Error       :
Name        :

Ending VM status: PowerState/running

Account ID of current context:
MSI@50342
```

9. Change the value for the `METHOD` parameter to `UA`.

10. Select **Start**. The runbook will attempt to change the power state of your VM using the named user-assigned managed identity. Once the runbook completes, the output should look similar to the following:

```
Using user-assigned managed identity

Beginning VM status: PowerState/running

OperationId : 679fcadf-d0b9-406a-9282-66bc211a9fbf
Status      : Succeeded
StartTime   : 8/3/2021 11:06:03 PM
EndTime     : 8/3/2021 11:06:49 PM
Error       :
Name        :

Ending VM status: PowerState/deallocated

Account ID of current context:
9034f5d3-c46d-44d4-afdf-c78aeab837ea
```

Clean up Resources

To remove any resources no longer needed, run the following runbook.

```
#Remove runbook
Remove-AzAutomationRunbook ` 
    -ResourceGroupName $resourceGroup ` 
    -AutomationAccountName $automationAccount ` 
    -Name "miTesting" ` 
    -Force

# Remove role assignments
Remove-AzRoleAssignment ` 
    -ObjectId $UAMI ` 
    -ResourceGroupName $resourceGroup ` 
    -RoleDefinitionName $role1

Remove-AzRoleAssignment ` 
    -ObjectId $SAMI ` 
    -ResourceGroupName $resourceGroup ` 
    -RoleDefinitionName $role2

Remove-AzRoleAssignment ` 
    -ObjectId $SAMI ` 
    -ResourceGroupName $resourceGroup ` 
    -RoleDefinitionName $role1
```

Next steps

In this tutorial, you created a [PowerShell runbook](#) in Azure Automation that used [managed identities](#), rather than the Run As account to interact with resources. For a look at PowerShell workflow runbooks, see:

[Tutorial: Create a PowerShell Workflow runbook](#)

Tutorial: Create a PowerShell Workflow runbook

9/3/2021 • 7 minutes to read • [Edit Online](#)

This tutorial walks you through the creation of a [PowerShell Workflow runbook](#) in Azure Automation.

PowerShell Workflow runbooks are text runbooks based on Windows PowerShell Workflow. You can create and edit the code of the runbook using the text editor in the Azure portal.

- Create a simple PowerShell Workflow runbook
- Test and publish the runbook
- Run and track the status of the runbook job
- Update the runbook to start an Azure virtual machine with runbook parameters

Prerequisites

To complete this tutorial, you need:

- Azure subscription. If you don't have one yet, you can [activate your MSDN subscriber benefits](#) or sign up for a [free account](#).
- [Automation account](#) to hold the runbook and authenticate to Azure resources. This account must have permission to start and stop the virtual machine.
- An Azure virtual machine. Since you stop and start this machine, it shouldn't be a production VM.

Step 1 - Create new runbook

Start by creating a simple runbook that outputs the text `Hello World`.

1. In the Azure portal, open your Automation account.

The Automation account page gives you a quick view of the resources in this account. You should already have some assets. Most of those assets are the modules automatically included in a new Automation account. You should also have the Credential asset associated with your subscription.

2. Select **Runbooks** under **Process Automation** to open the list of runbooks.
3. Create a new runbook by selecting **Create a runbook**.
4. Give the runbook the name **MyFirstRunbook-Workflow**.
5. In this case, you're going to create a [PowerShell Workflow runbook](#). Select **PowerShell Workflow** for **Runbook type**.
6. Click **Create** to create the runbook and open the textual editor.

Step 2 - Add code to the runbook

You can either type code directly into the runbook, or you can select cmdlets, runbooks, and assets from the Library control and add them to the runbook with any related parameters. For this tutorial, you type code directly into the runbook.

1. Your runbook is currently empty with only the required `Workflow` keyword, the name of the runbook, and the braces that encase the entire workflow.

```
Workflow MyFirstRunbook-Workflow
{
}
```

2. Type `Write-Output "Hello World"` between the braces.

```
Workflow MyFirstRunbook-Workflow
{
    Write-Output "Hello World"
}
```

3. Save the runbook by clicking **Save**.

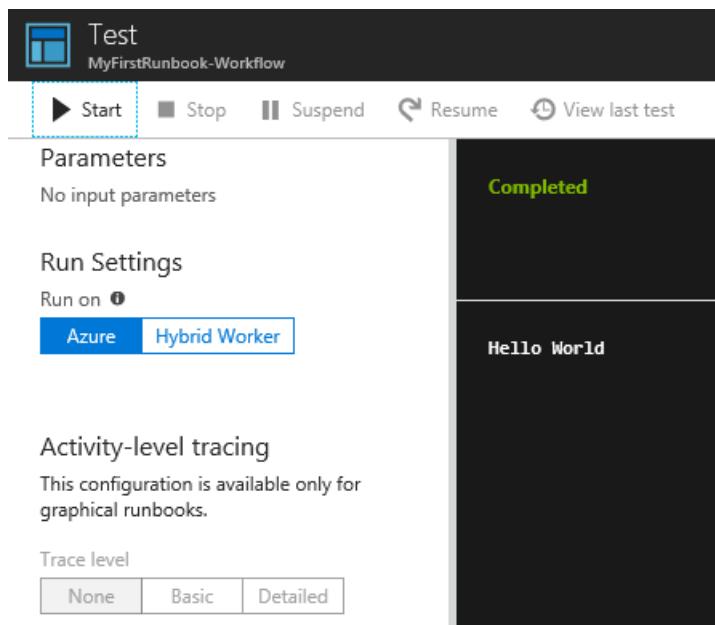
Step 3 - Test the runbook

Before you publish the runbook to make it available in production, you should test it to make sure that it works properly. Testing a runbook runs its Draft version and allows you to view its output interactively.

1. Select **Test pane** to open the Test pane.
2. Click **Start** to start the test, with testing the only enabled option.
3. Note that a [runbook job](#) is created and its status is displayed in the pane.

The job status starts as Queued, indicating that the job is waiting for a runbook worker in the cloud to become available. The status changes to Starting when a worker claims the job. Finally, the status becomes Running when the runbook actually starts to run.

4. When the runbook job completes, the Test pane displays its output. In this case, you see `Hello World`.



5. Close the Test pane to return to the canvas.

Step 4 - Publish and start the runbook

The runbook that you've created is still in Draft mode. You must publish it before you can run it in production. When you publish a runbook, you overwrite the existing Published version with the Draft version. In this case, you don't have a Published version yet because you just created the runbook.

1. Click **Publish** to publish the runbook and then **Yes** when prompted.

2. Scroll left to view the runbook in the Runbooks page and note that the **Authoring Status** field is set to **Published**.

3. Scroll back to the right to view the page for **MyFirstRunbook-Workflow**.

The options across the top allow you to start the runbook now, schedule a future start time, or create a [webhook](#) so that the runbook can be started through an HTTP call.

4. Select **Start** and then **Yes** when prompted to start the runbook.



5. A Job pane is opened for the runbook job that has been created. In this case, leave the pane open so you can watch the job's progress.

6. Note that the job status is shown in **Job Summary**. This status matches the statuses that you saw when testing the runbook.

A screenshot of the 'Job' pane for the runbook 'MyFirstRunbook-Workflow'.

Job Summary:

Job Id	3eb16ac6-09ff-4037-b67e-61b961c6ff20	Created	1/24/2018 1:33 PM
Job status	Completed	Last Update	1/24/2018 1:34 PM
Run As	User	Runbook	MyFirstRunbook-Workflow
Ran on	Azure	Source snapshot	View source snapshot

Overview:

Input	0	Output	0	All Logs
Errors	0	Warnings	0	

Exception:

None

7. Once the runbook status shows Completed, click **Output**. The Output page is opened, where you can see your `Hello World` message.



► Resume ■ Stop || Suspend

Essentials ^

Job Id	Created
3eb16ac6-09ff-4037-b67e-61b961c6ff20	1/24/2018 1:33 PM
Job status	Last Update
Completed	1/24/2018 1:34 PM
Run As	Runbook
User	MyFirstRunbook-Workflow
Ran on	Source snapshot
Azure	View source snapshot

Overview



Errors 0 ✗

Warnings 0 ⚠

Exception

None

8. Close the Output page.
9. Click All Logs to open the Streams pane for the runbook job. You should only see `Hello World` in the output stream. Note that the Streams pane can show other streams for a runbook job, such as verbose and error streams, if the runbook writes to them.

The screenshot shows the Azure Runbook Job pane for 'MyFirstRunbook-Workflow'. At the top, there are buttons for Resume, Stop, and Suspend. Below that is the 'Essentials' section with details like Job Id, Created, Job status, Last Update, Run As, Runbook, and Source snapshot. The 'Overview' section includes metrics for Input (0), Output (0), and All Logs (highlighted with a red box). It also shows 0 Errors and 0 Warnings. The 'Exception' section indicates 'None'.

10. Close the Streams pane and the Job pane to return to the MyFirstRunbook page.
11. Click **Jobs** under **Resources** to open the Jobs page for this runbook. This page lists all the jobs created by your runbook. You should only see one job listed, since you have run the job only once.

The screenshot shows the 'Details' section of the Azure Resource Groups page. The 'Jobs' section is highlighted with a red box. It displays a count of 0 and a clock icon, indicating no scheduled jobs.

12. Click the job name to open the same Job pane that you viewed when you started the runbook. Use this pane to view the details of any job created for the runbook.

Step 5 - Add authentication to manage Azure resources

You've tested and published your runbook, but so far it doesn't do anything useful. You want to have it manage Azure resources. It can't do that unless it authenticates using the credentials for the subscription. Authentication uses the [Connect-AzAccount](#) cmdlet.

NOTE

For PowerShell runbooks, `Add-AzAccount` and `Add-AzureRMAccount` are aliases for `Connect-AzAccount`. You can use these cmdlets or you can [update your modules](#) in your Automation account to the latest versions. You might need to update your modules even if you have just created a new Automation account.

1. Navigate to the MyFirstRunbook-Workflow page and open the textual editor by clicking **Edit**.
2. Delete the `Write-Output` line.
3. Position the cursor on a blank line between the braces.
4. Type or copy and paste the following code, which handles the authentication with your Automation Run As account.

```
# Ensures you do not inherit an AzContext in your runbook
Disable-AzContextAutosave -Scope Process

$Conn = Get-AutomationConnection -Name AzureRunAsConnection
Connect-AzAccount -ServicePrincipal -Tenant $Conn.TenantID ` 
-ApplicationId $Conn.ApplicationID -CertificateThumbprint $Conn.CertificateThumbprint

$AzureContext = Select-AzSubscription -SubscriptionId $Conn.SubscriptionID
```

5. Click **Test pane** so that you can test the runbook.
6. Click **Start** to start the test. Once it completes, you should see output similar to the following, displaying basic information from your account. This action confirms that the credential is valid.



The screenshot shows the 'Test pane' results for a runbook. At the top, a green bar indicates the status is 'Completed'. Below this, there are two sections: 'Environments' and 'Context'. The 'Environments' section lists three environments: 'AzureCloud', 'AzureCloud', and 'AzureChinaCloud'. The 'Context' section shows the 'Microsoft.Azure' context with a note about multiple subscriptions.

Environments	Context
[[AzureCloud, AzureCloud], [AzureChinaCloud, AzureChinaCloud], [AzureUSGovernment, AzureUSGovernment]]} Microsoft.Azure...	

Step 6 - Add code to start a virtual machine

Now that your runbook is authenticating to the Azure subscription, you can manage resources. Let's add a command to start a virtual machine. You can pick any VM in your Azure subscription, and for now you're hardcoding that name in the runbook. If you're managing resources across multiple subscriptions, you need to use the `AzContext` parameter with the [Get-AzContext](#) cmdlet.

1. Provide the name and resource group name of the VM to start by entering a call to the [Start-AzVM](#) cmdlet as shown below.

```

workflow MyFirstRunbook-Workflow
{
    # Ensures that you do not inherit an AzContext in your runbook
    Disable-AzContextAutosave -Scope Process

    $Conn = Get-AutomationConnection -Name AzureRunAsConnection
    Connect-AzAccount -ServicePrincipal -Tenant $Conn.TenantID -ApplicationId $Conn.ApplicationID -
    CertificateThumbprint $Conn.CertificateThumbprint

    $AzureContext = Get-AzSubscription -SubscriptionId $Conn.SubscriptionID

    Start-AzVM -Name 'VMName' -ResourceGroupName 'ResourceGroupName' -AzContext $AzureContext
}

```

2. Save the runbook and then click **Test pane** so that you can test it.
3. Click **Start** to start the test. Once it completes, check that the VM has been started.

Step 7 - Add an input parameter to the runbook

Your runbook currently starts the VM that you have hardcoded in the runbook. It will be more useful if you can specify the VM when the runbook is started. Let's add input parameters to the runbook to provide that functionality.

1. Add variables for the `VMName` and `ResourceGroupName` parameters to the runbook, and use the variables with the `Start-AzVM` cmdlet as shown below.

```

workflow MyFirstRunbook-Workflow
{
    Param(
        [string]$VMName,
        [string]$ResourceGroupName
    )
    # Ensures you do not inherit an AzContext in your runbook
    Disable-AzContextAutosave -Scope Process

    $Conn = Get-AutomationConnection -Name AzureRunAsConnection
    Connect-AzAccount -ServicePrincipal -Tenant $Conn.TenantID -ApplicationId $Conn.ApplicationID -
    CertificateThumbprint $Conn.CertificateThumbprint
    Start-AzVM -Name $VMName -ResourceGroupName $ResourceGroupName
}

```

2. Save the runbook and open the Test pane. You can now provide values for the two input variables that are in the test.
3. Close the Test pane.
4. Click **Publish** to publish the new version of the runbook.
5. Stop the VM that you have started.
6. Click **Start** to start the runbook.
7. Type in the values for **VMNAME** and **RESOURCEGROUPNAME** for the VM that you're going to start.

Parameters

VMNAME

ContosoVM1

Optional, String

RESOURCEGROUPNAME

ContosoVM

Optional, String

Run Settings

Run on

Azure

Hybrid Worker

- When the runbook completes, verify that the VM has started.

Next steps

In this tutorial, you created a PowerShell workflow runbook. For a look at Python 3 runbooks, see:

[Tutorial: Create a Python 3 runbook \(preview\)](#)

Tutorial: Create a Python 3 runbook (preview)

4/28/2021 • 7 minutes to read • [Edit Online](#)

This tutorial walks you through the creation of a [Python 3 runbook](#) (preview) in Azure Automation. Python runbooks compile under Python 2 and 3. You can directly edit the code of the runbook using the text editor in the Azure portal.

- Create a simple Python runbook
- Test and publish the runbook
- Run and track the status of the runbook job
- Update the runbook to start an Azure virtual machine with runbook parameters

Prerequisites

To complete this tutorial, you need the following:

- Azure subscription. If you don't have one yet, you can [activate your MSDN subscriber benefits](#) or sign up for a [free account](#).
- [Automation account](#) to hold the runbook and authenticate to Azure resources. This account must have permission to start and stop the virtual machine. The [Run As account](#) is required for this tutorial.
- An Azure virtual machine. During this tutorial, you will start and stop this machine, so it should not be a production VM.

Create a new runbook

You start by creating a simple runbook that outputs the text *Hello World*.

1. In the Azure portal, open your Automation account.

The Automation account page gives you a quick view of the resources in this account. You should already have some assets. Most of those assets are the modules that are automatically included in a new Automation account. You should also have the Run As account credential asset that's mentioned in the [prerequisites](#).

2. Select **Runbooks** under **Process Automation** to open the list of runbooks.
3. Select **Add a runbook** to create a new runbook.
4. Give the runbook the name **MyFirstRunbook-Python**.
5. Select **Python 3** for **Runbook type**.
6. Select **Create** to create the runbook and open the textual editor.

Add code to the runbook

Now you add a simple command to print the text `Hello World`.

```
print("Hello World!")
```

Select **Save** to save the runbook.

Test the runbook

Before you publish the runbook to make it available in production, you want to test it to make sure that it works properly. When you test a runbook, you run its draft version and view its output interactively.

1. Select **Test pane** to open the **Test** pane.
2. Select **Start** to start the test. This should be the only enabled option.
3. A **runbook job** is created and its status displayed. The job status starts as **Queued**, indicating that it is waiting for a runbook worker in the cloud to become available. It changes to **Starting** when a worker claims the job, and then **Running** when the runbook actually starts running.
4. When the runbook job completes, its output is displayed. In this case, you should see `Hello World`.
5. Close the **Test** pane to return to the canvas.

Publish and start the runbook

The runbook that you created is still in draft mode. You need to publish it before you can run it in production. When you publish a runbook, you overwrite the existing published version with the draft version. In this case, you don't have a published version yet because you just created the runbook.

1. Select **Publish** to publish the runbook and then **Yes** when prompted.
2. If you scroll left to view the runbook on the **Runbooks** page, you should see an **Authoring Status** of **Published**.
3. Scroll back to the right to view the pane for **MyFirstRunbook-Python3**.
The options across the top allow you to start the runbook, view the runbook, or schedule it to start at some time in the future.
4. Select **Start** and then select **OK** when the **Start Runbook** pane opens.
5. A **Job** pane is opened for the runbook job that you created. You can close this pane, but let's leave it open so that you can watch the job's progress.
6. The job status is shown in **Job Summary** and matches the statuses that you saw when you tested the runbook.
7. Once the runbook status shows **Completed**, select **Output**. The **Output** pane is opened, where you can see `Hello World`.
8. Close the **Output** pane.
9. Select **All Logs** to open the **Streams** pane for the runbook job. You should only see `Hello World` in the Output stream. However, this pane can show other streams for a runbook job, such as **Verbose** and **Error**, if the runbook writes to them.
10. Close the **Streams** pane and the **Job** pane to return to the **MyFirstRunbook-Python3** pane.
11. Select **Jobs** to open the **Jobs** page for this runbook. This page lists all jobs created by this runbook. You should only see one job listed since you only ran the job once.
12. You can select this job to open the same **Job** pane that you viewed when you started the runbook. This pane allows you to go back in time and view the details of any job that was created for a particular runbook.

Add authentication to manage Azure resources

You've tested and published your runbook, but so far it doesn't do anything useful. You want to have it manage Azure resources. To do this, the script has to authenticate using the Run As account credential from your Automation account.

NOTE

The Automation account must have been created with the Run As account for there to be a Run As certificate. If your Automation account was not created with the Run As account, you can authenticate as described in [Authenticate with the Azure Management Libraries for Python](#) or [create a Run As account](#).

1. Open the textual editor by selecting **Edit** on the **MyFirstRunbook-Python3** pane.
2. Add the following code to authenticate to Azure:

```
import os
from azure.mgmt.compute import ComputeManagementClient
import azure.mgmt.resource
import automationassets

def get_automation_runas_credential(runas_connection):
    from OpenSSL import crypto
    import binascii
    from msrestazure import azure_active_directory
    import adal

    # Get the Azure Automation RunAs service principal certificate
    cert = automationassets.get_automation_certificate("AzureRunAsCertificate")
    pks12_cert = crypto.load_pkcs12(cert)
    pem_pkey = crypto.dump_privatekey(crypto.FILETYPE_PEM,pks12_cert.get_privatekey())

    # Get run as connection information for the Azure Automation service principal
    application_id = runas_connection["ApplicationId"]
    thumbprint = runas_connection["CertificateThumbprint"]
    tenant_id = runas_connection["TenantId"]

    # Authenticate with service principal certificate
    resource ="https://management.core.windows.net/"
    authority_url = ("https://login.microsoftonline.com/" +tenant_id)
    context = adal.AuthenticationContext(authority_url)
    return azure_active_directory.AdalAuthentication(
        lambda: context.acquire_token_with_client_certificate(
            resource,
            application_id,
            pem_pkey,
            thumbprint)
    )

    # Authenticate to Azure using the Azure Automation RunAs service principal
    runas_connection = automationassets.get_automation_connection("AzureRunAsConnection")
    azure_credential = get_automation_runas_credential(runas_connection)
```

Add code to create Python Compute client and start the VM

To work with Azure VMs, create an instance of the [Azure Compute client for Python](#).

Use the compute client to start the VM. Add the following code to the runbook:

```

# Initialize the compute management client with the Run As credential and specify the subscription to work
against.
compute_client = ComputeManagementClient(
    azure_credential,
    str(runas_connection["SubscriptionId"])
)

print('\nStart VM')
async_vm_start = compute_client.virtual_machines.start(
    "MyResourceGroup", "TestVM")
async_vm_start.wait()

```

Where `MyResourceGroup` is the name of the resource group that contains the VM, and `TestVM` is the name of the VM that you want to start.

Test and run the runbook again to see that it starts the VM.

Use input parameters

The runbook currently uses hard-coded values for the names of the resource group and the VM. Now let's add code that gets these values from input parameters.

You use the `sys.argv` variable to get the parameter values. Add the following code to the runbook immediately after the other `import` statements:

```

import sys

resource_group_name = str(sys.argv[1])
vm_name = str(sys.argv[2])

```

This imports the `sys` module, and creates two variables to hold the resource group and VM names. Notice that the element of the argument list, `sys.argv[0]`, is the name of the script, and is not input by the user.

Now you can modify the last two lines of the runbook to use the input parameter values instead of using hard-coded values:

```

async_vm_start = compute_client.virtual_machines.start(
    resource_group_name, vm_name)
async_vm_start.wait()

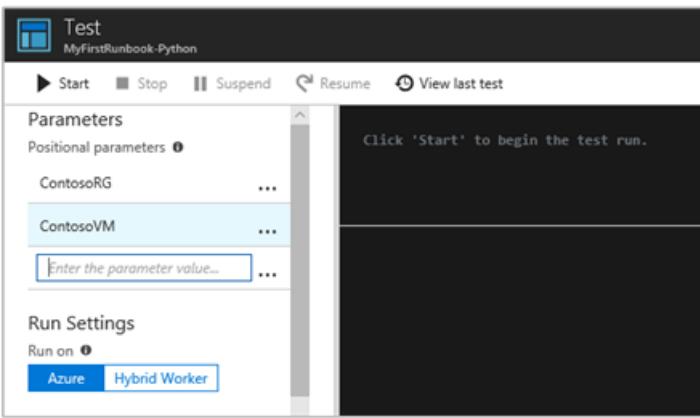
```

When you start a Python runbook, either from the **Test** pane or as a published runbook, you can enter the values for parameters in the **Start Runbook** page under **Parameters**.

After you start entering a value in the first box, a second appears, and so on, so that you can enter as many parameter values as necessary.

The values are available to the script in the `sys.argv` array as in the code you just added.

Enter the name of your resource group as the value for the first parameter, and the name of the VM to start as the value of the second parameter.



Select OK to start the runbook. The runbook runs and starts the VM that you specified.

Error handling in Python

You can also use the following conventions to retrieve various streams from your Python runbooks, including WARNING, ERROR, and DEBUG streams.

```
print("Hello World output")
print("ERROR: - Hello world error")
print("WARNING: - Hello world warning")
print("DEBUG: - Hello world debug")
print("VERBOSE: - Hello world verbose")
```

The following example shows this convention used in a `try...except` block.

```
try:
    raise Exception('one', 'two')
except Exception as detail:
    print ('ERROR: Handling run-time error:', detail)
```

Next steps

- To learn more about runbook types, their advantages and limitations, see [Azure Automation runbook types](#).
- To learn about developing for Azure with Python, see [Azure for Python developers](#).
- To view sample Python 3 runbooks, see the [Azure Automation GitHub](#) repository.

Azure Automation account authentication overview

9/10/2021 • 9 minutes to read • [Edit Online](#)

Azure Automation allows you to automate tasks against resources in Azure, on-premises, and with other cloud providers such as Amazon Web Services (AWS). You can use runbooks to automate your tasks, or a Hybrid Runbook Worker if you have business or operational processes to manage outside of Azure. Working in any one of these environments require permissions to securely access the resources with the minimal rights required.

This article covers authentication scenarios supported by Azure Automation and tells how to get started based on the environment or environments that you need to manage.

Automation account

When you start Azure Automation for the first time, you must create at least one Automation account. Automation accounts allow you to isolate your Automation resources, runbooks, assets, and configurations from the resources of other accounts. You can use Automation accounts to separate resources into separate logical environments or delegated responsibilities. For example, you might use one account for development, another for production, and another for your on-premises environment. Or you might dedicate an Automation account to manage operating system updates across all of your machines with [Update Management](#).

An Azure Automation account is different from your Microsoft account or accounts created in your Azure subscription. For an introduction to creating an Automation account, see [Create an Automation account](#).

Automation resources

The Automation resources for each Automation account are associated with a single Azure region, but the account can manage all the resources in your Azure subscription. The main reason to create Automation accounts in different regions is if you have policies that require data and resources to be isolated to a specific region.

All tasks that you create against resources using Azure Resource Manager and the PowerShell cmdlets in Azure Automation must authenticate to Azure using Azure Active Directory (Azure AD) organizational identity credential-based authentication.

Managed identities (preview)

A managed identity from Azure Active Directory (Azure AD) allows your runbook to easily access other Azure AD-protected resources. The identity is managed by the Azure platform and doesn't require you to provision or rotate any secrets. For more information about managed identities in Azure AD, see [Managed identities for Azure resources](#).

Here are some of the benefits of using managed identities:

- Using a managed identity instead of the Automation Run As account makes management simpler. You don't have to renew the certificate used by a Run As account.
- Managed identities can be used without any additional cost.
- You don't have to renew the certificate used by the Automation Run As account.
- You don't have to specify the Run As connection object in your runbook code. You can access resources using your Automation account's managed identity from a runbook without creating certificates, connections, Run As accounts, etc.

An Automation account can be granted two types of identities:

- A system-assigned identity is tied to your application and is deleted if your app is deleted. An app can only have one system-assigned identity.
- A user-assigned identity is a standalone Azure resource that can be assigned to your app. An app can have multiple user-assigned identities.

NOTE

User assigned identities are supported for cloud jobs only. To learn more about the different managed identities, see [Manage identity types](#).

For details on using managed identities, see [Enable managed identity for Azure Automation \(preview\)](#).

Run As accounts

Run As accounts in Azure Automation provide authentication for managing Azure Resource Manager resources or resources deployed on the classic deployment model. There are two types of Run As accounts in Azure Automation:

To create or renew a Run As account, permissions are needed at three levels:

- Subscription,
- Azure Active Directory (Azure AD), and
- Automation account

Subscription permissions

You need the `Microsoft.Authorization/*/Write` permission. This permission is obtained through membership of one of the following Azure built-in roles:

- [Owner](#)
- [User Access Administrator](#)

To configure or renew Classic Run As accounts, you must have the Co-administrator role at the subscription level. To learn more about classic subscription permissions, see [Azure classic subscription administrators](#).

Azure AD permissions

To be able to create or renew the service principal, you need to be a member of one of the following Azure AD built-in roles:

- [Application Administrator](#)
- [Application Developer](#)

Membership can be assigned to **ALL** users in the tenant at the directory level, which is the default behavior. You can grant membership to either role at the directory level. For more information, see [Who has permission to add applications to my Azure AD instance?](#)

Automation account permissions

To be able to create or update the Automation account, you need to be a member of one of the following Automation account roles:

- [Owner](#)
- [Contributor](#)
- [Custom Azure Automation Contributor](#)

To learn more about the Azure Resource Manager and Classic deployment models, see [Resource Manager and classic deployment](#).

NOTE

Azure Cloud Solution Provider (CSP) subscriptions support only the Azure Resource Manager model. Non-Azure Resource Manager services are not available in the program. When you are using a CSP subscription, the Azure Classic Run As account is not created, but the Azure Run As account is created. To learn more about CSP subscriptions, see [Available services in CSP subscriptions](#).

When you create an Automation account, the Run As account is created by default at the same time with a self-signed certificate. If you chose not to create it along with the Automation account, it can be created individually at a later time. An Azure Classic Run As Account is optional, and is created separately if you need to manage classic resources.

If you want to use a certificate issued by your enterprise or third-party certification authority (CA) instead of the default self-signed certificate, can use the [PowerShell script to create a Run As account](#) option for your Run As and Classic Run As accounts.

Run As account

When you create a Run As account, it performs the following tasks:

- Creates an Azure AD application with a self-signed certificate, creates a service principal account for the application in Azure AD, and assigns the [Contributor](#) role for the account in your current subscription. You can change the certificate setting to [Reader](#) or any other role. For more information, see [Role-based access control in Azure Automation](#).
- Creates an Automation certificate asset named `AzureRunAsCertificate` in the specified Automation account. The certificate asset holds the certificate private key that the Azure AD application uses.
- Creates an Automation connection asset named `AzureRunAsConnection` in the specified Automation account. The connection asset holds the application ID, tenant ID, subscription ID, and certificate thumbprint.

Azure Classic Run As account

When you create an Azure Classic Run As account, it performs the following tasks:

NOTE

You must be a co-administrator on the subscription to create or renew this type of Run As account.

- Creates a management certificate in the subscription.
- Creates an Automation certificate asset named `AzureClassicRunAsCertificate` in the specified Automation account. The certificate asset holds the certificate private key used by the management certificate.
- Creates an Automation connection asset named `AzureClassicRunAsConnection` in the specified Automation account. The connection asset holds the subscription name, subscription ID, and certificate asset name.

Service principal for Run As account

The service principal for a Run As account doesn't have permissions to read Azure AD by default. If you want to

add permissions to read or manage Azure AD, you must grant the permissions on the service principal under **API permissions**. To learn more, see [Add permissions to access your web API](#).

Run As account permissions

This section defines permissions for both regular Run As accounts and Classic Run As accounts.

- To create or update a Run As account, an Application administrator in Azure Active Directory and an Owner in the subscription can complete all the tasks.
- To configure or renew Classic Run As accounts, you must have the Co-administrator role at the subscription level. To learn more about classic subscription permissions, see [Azure classic subscription administrators](#).

In a situation where you have separation of duties, the following table shows a listing of the tasks, the equivalent cmdlet, and permissions needed:

TASK	CMDLET	MINIMUM PERMISSIONS	WHERE YOU SET THE PERMISSIONS
Create Azure AD Application	New-AzADApplication	Application Developer role ¹	Azure AD Home > Azure AD > App Registrations
Add a credential to the application.	New-AzADAppCredential	Application Administrator or Global Administrator ¹	Azure AD Home > Azure AD > App Registrations
Create and get an Azure AD service principal	New-AzADServicePrincipal Get-AzADServicePrincipal	Application Administrator or Global Administrator ¹	Azure AD Home > Azure AD > App Registrations
Assign or get the Azure role for the specified principal	New-AzRoleAssignment Get-AzRoleAssignment	User Access Administrator or Owner, or have the following permissions: <small>Microsoft.Authorization/operations Microsoft.Authorization/permissions/read Microsoft.Authorization/roleDefinitions/read Microsoft.Authorization/roleAssignments/write Microsoft.Authorization/roleAssignments/read Microsoft.Authorization/roleAssignments/delete</small>	Subscription Home > Subscriptions > <subscription name> - Access Control (IAM)
Create or remove an Automation certificate	New-AzAutomationCertificate Remove-AzAutomationCertificate	Contributor on resource group	Automation account resource group
Create or remove an Automation connection	New-AzAutomationConnection Remove-AzAutomationConnection	Contributor on resource group	Automation account resource group

¹ Non-administrator users in your Azure AD tenant can [register AD applications](#) if the Azure AD tenant's **Users can register applications** option on the **User settings** page is set to **Yes**. If the application registration setting is **No**, the user performing this action must be as defined in this table.

If you aren't a member of the subscription's Active Directory instance before you're added to the Global

Administrator role of the subscription, you're added as a guest. In this situation, you receive a

You do not have permissions to create... warning on the **Add Automation account** page.

To verify that the situation producing the error message has been remedied:

1. From the Azure Active Directory pane in the Azure portal, select **Users and groups**.
2. Select **All users**.
3. Choose your name, then select **Profile**.
4. Ensure that the value of the **User type** attribute under your user's profile isn't set to **Guest**.

Role-based access control

Role-based access control is available with Azure Resource Manager to grant permitted actions to an Azure AD user account and Run As account, and authenticate the service principal. Read [Role-based access control in Azure Automation article](#) for further information to help develop your model for managing Automation permissions.

If you have strict security controls for permission assignment in resource groups, you need to assign the Run As account membership to the **Contributor** role in the resource group.

NOTE

We recommend you don't use the **Log Analytics Contributor** role to execute Automation jobs. Instead, create the Azure Automation Contributor custom role and use it for actions related to the Automation account. For more information, see [Custom Azure Automation Contributor role](#).

Runbook authentication with Hybrid Runbook Worker

Runbooks running on a Hybrid Runbook Worker in your datacenter or against computing services in other cloud environments like AWS, can't use the same method that is typically used for runbooks authenticating to Azure resources. This is because those resources are running outside of Azure and therefore, requires their own security credentials defined in Automation to authenticate to resources that they access locally. For more information about runbook authentication with runbook workers, see [Run runbooks on a Hybrid Runbook Worker](#).

For runbooks that use Hybrid Runbook Workers on Azure VMs, you can use [runbook authentication with managed identities](#) instead of Run As accounts to authenticate to your Azure resources.

Next steps

- To create an Automation account from the Azure portal, see [Create a standalone Azure Automation account](#).
- If you prefer to create your account using a template, see [Create an Automation account using an Azure Resource Manager template](#).
- For authentication using Amazon Web Services, see [Authenticate runbooks with Amazon Web Services](#).
- For a list of Azure services that support the managed identities for Azure resources feature, see [Services that support managed identities for Azure resources](#).

Runbook execution in Azure Automation

9/10/2021 • 15 minutes to read • [Edit Online](#)

Process automation in Azure Automation allows you to create and manage PowerShell, PowerShell Workflow, and graphical runbooks. For details, see [Azure Automation runbooks](#).

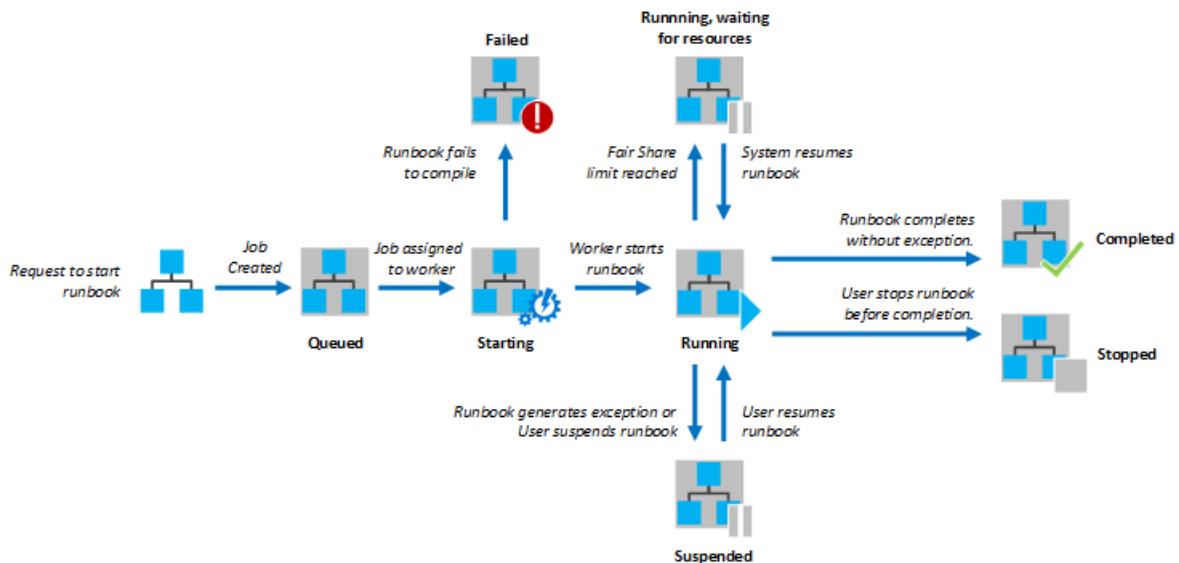
Automation executes your runbooks based on the logic defined inside them. If a runbook is interrupted, it restarts at the beginning. This behavior requires you to write runbooks that support being restarted if transient issues occur.

Starting a runbook in Azure Automation creates a job, which is a single execution instance of the runbook. Each job accesses Azure resources by making a connection to your Azure subscription. The job can only access resources in your datacenter if those resources are accessible from the public cloud.

Azure Automation assigns a worker to run each job during runbook execution. While workers are shared by many Automation accounts, jobs from different Automation accounts are isolated from one another. You can't control which worker services your job requests.

When you view the list of runbooks in the Azure portal, it shows the status of each job that has been started for each runbook. Azure Automation stores job logs for a maximum of 30 days.

The following diagram shows the lifecycle of a runbook job for [PowerShell runbooks](#), [PowerShell Workflow runbooks](#), and [graphical runbooks](#).



NOTE

For information about viewing or deleting personal data, see [Azure Data Subject Requests for the GDPR](#). For more information about GDPR, see the [GDPR section of the Microsoft Trust Center](#) and the [GDPR section of the Service Trust portal](#).

Runbook execution environment

Runbooks in Azure Automation can run on either an Azure sandbox or a [Hybrid Runbook Worker](#).

When runbooks are designed to authenticate and run against resources in Azure, they run in an Azure sandbox. Azure Automation assigns a worker to run each job during runbook execution in the sandbox. While workers are

shared by many Automation accounts, jobs from different Automation accounts are isolated from one another. Jobs using the same sandbox are bound by the resource limitations of the sandbox. The Azure sandbox environment does not support interactive operations. It prevents access to all out-of-process COM servers, and it does not support making [WMI calls](#) to the Win32 provider in your runbook. These scenarios are only supported by running the runbook on a Windows Hybrid Runbook Worker.

You can also use a [Hybrid Runbook Worker](#) to run runbooks directly on the computer that hosts the role and against local resources in the environment. Azure Automation stores and manages runbooks and then delivers them to one or more assigned computers.

Enabling the Azure Firewall on [Azure Storage](#), [Azure Key Vault](#), or [Azure SQL](#) blocks access from Azure Automation runbooks for those services. Access will be blocked even when the firewall exception to allow trusted Microsoft services is enabled, as Automation is not a part of the trusted services list. With an enabled firewall, access can only be made by using a Hybrid Runbook Worker and a [virtual network service endpoint](#).

NOTE

To run on a Linux Hybrid Runbook Worker, your scripts must be signed and the worker configured accordingly. Alternatively, [signature validation must be turned off](#).

The following table lists some runbook execution tasks with the recommended execution environment listed for each.

TASK	RECOMMENDATION	NOTES
Integrate with Azure resources	Azure Sandbox	Hosted in Azure, authentication is simpler. If you're using a Hybrid Runbook Worker on an Azure VM, you can use runbook authentication with managed identities .
Obtain optimal performance to manage Azure resources	Azure Sandbox	Script is run in the same environment, which has less latency.
Minimize operational costs	Azure Sandbox	There is no compute overhead and no need for a VM.
Execute long-running script	Hybrid Runbook Worker	Azure sandboxes have resource limits .
Interact with local services	Hybrid Runbook Worker	Directly access the host machine, or resources in other cloud environments or the on-premises environment.
Require third-party software and executables	Hybrid Runbook Worker	You manage the operating system and can install software.
Monitor a file or folder with a runbook	Hybrid Runbook Worker	Use a Watcher task on a Hybrid Runbook Worker.
Run a resource-intensive script	Hybrid Runbook Worker	Azure sandboxes have resource limits .
Use modules with specific requirements	Hybrid Runbook Worker	Some examples are: WinSCP - dependency on winscp.exe IIS administration - dependency on enabling or managing IIS

TASK	RECOMMENDATION	NOTES
Install a module with an installer	Hybrid Runbook Worker	Modules for sandbox must support copying.
Use runbooks or modules that require .NET Framework version different from 4.7.2	Hybrid Runbook Worker	Azure sandboxes support .NET Framework 4.7.2, and upgrading to a different version is not supported.
Run scripts that require elevation	Hybrid Runbook Worker	Sandboxes don't allow elevation. With a Hybrid Runbook Worker, you can turn off UAC and use Invoke-Command when running the command that requires elevation.
Run scripts that require access to Windows Management Instrumentation (WMI)	Hybrid Runbook Worker	Jobs running in sandboxes in the cloud can't access WMI provider.

Temporary storage in a sandbox

If you need to create temporary files as part of your runbook logic, you can use the Temp folder (that is, `$env:TEMP`) in the Azure sandbox for runbooks running in Azure. The only limitation is you cannot use more than 1 GB of disk space, which is the quota for each sandbox. When working with PowerShell workflows, this scenario can cause a problem because PowerShell workflows use checkpoints and the script could be retried in a different sandbox.

With the hybrid sandbox, you can use `c:\temp` based on the availability of storage on a Hybrid Runbook Worker. However, per Azure VM recommendations, you should not use the [temporary disk](#) on Windows or Linux for data that needs to be persisted.

Resources

Your runbooks must include logic to deal with [resources](#), for example, VMs, the network, and resources on the network. Resources are tied to an Azure subscription, and runbooks require appropriate credentials to access any resource. For an example of handling resources in a runbook, see [Handle resources](#).

Security

Azure Automation uses the [Azure Security Center \(ASC\)](#) to provide security for your resources and detect compromise in Linux systems. Security is provided across your workloads, whether resources are in Azure or not. See [Introduction to authentication in Azure Automation](#).

ASC places constraints on users who can run any scripts, either signed or unsigned, on a VM. If you are a user with root access to a VM, you must explicitly configure the machine with a digital signature or turn it off. Otherwise, you can only run a script to apply operating system updates after creating an Automation account and enabling the appropriate feature.

Subscriptions

An Azure [subscription](#) is an agreement with Microsoft to use one or more cloud-based services, for which you are charged. For Azure Automation, each subscription is linked to an Azure Automation account, and you can [create multiple subscriptions](#) in the account.

Credentials

A runbook requires appropriate [credentials](#) to access any resource, whether for Azure or third-party systems. These credentials are stored in Azure Automation, Key Vault, etc.

Azure Monitor

Azure Automation makes use of [Azure Monitor](#) for monitoring its machine operations. The operations require a Log Analytics workspace and a [Log Analytics agent](#).

Log Analytics agent for Windows

The [Log Analytics agent for Windows](#) works with Azure Monitor to manage Windows VMs and physical computers. The machines can be running either in Azure or in a non-Azure environment, such as a local datacenter.

NOTE

The Log Analytics agent for Windows was previously known as the Microsoft Monitoring Agent (MMA).

Log Analytics agent for Linux

The [Log Analytics agent for Linux](#) works similarly to the agent for Windows, but connects Linux computers to Azure Monitor. The agent is installed with a **nxautomation** user account that allows execution of commands requiring root permissions, for example, on a Hybrid Runbook Worker. The **nxautomation** account is a system account that doesn't require a password.

The **nxautomation** account with the corresponding sudo permissions must be present during [installation of a Linux Hybrid Runbook worker](#). If you try to install the worker and the account is not present or doesn't have the appropriate permissions, the installation fails.

Do not change the permissions of the `sudoers.d` folder or its ownership. Sudo permission is required for the **nxautomation** account and the permissions should not be removed. Restricting this to certain folders or commands may result in a breaking change.

The logs available for the Log Analytics agent and the **nxautomation** account are:

- `/var/opt/microsoft/omsagent/log/omsagent.log` - Log Analytics agent log
- `/var/opt/microsoft/omsagent/run/automationworker/worker.log` - Automation worker log

NOTE

The **nxautomation** user enabled as part of Update Management executes only signed runbooks.

Runbook permissions

A runbook needs permissions for authentication to Azure, through credentials. See [Azure Automation authentication overview](#).

Modules

Azure Automation includes the following PowerShell modules:

- `OrchestratorAssetManagement.Cmdlets` - contains several internal cmdlets that are only available when you execute runbooks in the Azure sandbox environment or on a Windows Hybrid Runbook Worker. These cmdlets are designed to be used instead of Azure PowerShell cmdlets to interact with your Automation

account resources.

- Az.Automation - the recommended PowerShell module for interacting with Azure Automation that replaces the AzureRM Automation module. The Az.Automation module is not automatically included when you create an Automation account and you need to import them manually.
- AzureRM.Automation - installed by default when you create an Automation account.

Also supported are installable modules, based on the cmdlets that your runbooks and DSC configurations require. For details of the modules that are available for your runbooks and DSC configurations, see [Manage modules in Azure Automation](#).

Certificates

Azure Automation uses [certificates](#) for authentication to Azure or adds them to Azure or third-party resources. The certificates are stored securely for access by runbooks and DSC configurations.

Your runbooks can use self-signed certificates, which are not signed by a certificate authority (CA). See [Create a new certificate](#).

Jobs

Azure Automation supports an environment to run jobs from the same Automation account. A single runbook can have many jobs running at one time. The more jobs you run at the same time, the more often they can be dispatched to the same sandbox.

Jobs running in the same sandbox process can affect each other. One example is running the [Disconnect-AzAccount](#) cmdlet. Execution of this cmdlet disconnects each runbook job in the shared sandbox process. For an example of working with this scenario, see [Prevent concurrent jobs](#).

NOTE

PowerShell jobs started from a runbook that runs in an Azure sandbox might not run in the full [PowerShell language mode](#).

Job statuses

The following table describes the statuses that are possible for a job. You can view a status summary for all runbook jobs or drill into details of a specific runbook job in the Azure portal. You can also configure integration with your Log Analytics workspace to forward runbook job status and job streams. For more information about integrating with Azure Monitor logs, see [Forward job status and job streams from Automation to Azure Monitor logs](#). See also [Obtain job statuses](#) for an example of working with statuses in a runbook.

STATUS	DESCRIPTION
Activating	The job is being activated.
Completed	The job completed successfully.
Failed	A graphical or PowerShell Workflow runbook failed to compile. A PowerShell runbook failed to start or the job had an exception. See Azure Automation runbook types .
Failed, waiting for resources	The job failed because it reached the fair share limit three times and started from the same checkpoint or from the start of the runbook each time.

STATUS	DESCRIPTION
Queued	The job is waiting for resources on an Automation worker to become available so that it can be started.
Resuming	The system is resuming the job after it was suspended.
Running	The job is running.
Running, waiting for resources	The job has been unloaded because it reached the fair share limit. It will resume shortly from its last checkpoint.
Starting	The job has been assigned to a worker, and the system is starting it.
Stopped	The job was stopped by the user before it was completed.
Stopping	The system is stopping the job.
Suspended	Applies to graphical and PowerShell Workflow runbooks only. The job was suspended by the user, by the system, or by a command in the runbook. If a runbook doesn't have a checkpoint, it starts from the beginning. If it has a checkpoint, it can start again and resume from its last checkpoint. The system only suspends the runbook when an exception occurs. By default, the <code>ErrorActionPreference</code> variable is set to Continue, indicating that the job keeps running on an error. If the preference variable is set to Stop, the job suspends on an error.
Suspending	Applies to graphical and PowerShell Workflow runbooks only. The system is trying to suspend the job at the request of the user. The runbook must reach its next checkpoint before it can be suspended. If it has already passed its last checkpoint, it completes before it can be suspended.

Activity logging

Execution of runbooks in Azure Automation writes details in an activity log for the Automation account. For details of using the log, see [Retrieve details from Activity log](#).

Exceptions

This section describes some ways to handle exceptions or intermittent issues in your runbooks. An example is a WebSocket exception. Correct exception handling prevents transient network failures from causing your runbooks to fail.

ErrorActionPreference

The `ErrorActionPreference` variable determines how PowerShell responds to a non-terminating error.

Terminating errors always terminate and are not affected by `ErrorActionPreference`.

When the runbook uses `ErrorActionPreference`, a normally non-terminating error such as `PathNotFound` from the [Get-ChildItem](#) cmdlet stops the runbook from completing. The following example shows the use of `ErrorActionPreference`. The final [Write-Output](#) command never executes, as the script stops.

```
$ErrorActionPreference = 'Stop'  
Get-ChildItem -path nofile.txt  
Write-Output "This message will not show"
```

Try Catch Finally

[Try Catch Finally](#) is used in PowerShell scripts to handle terminating errors. The script can use this mechanism to catch specific exceptions or general exceptions. The `catch` statement should be used to track or try to handle errors. The following example tries to download a file that does not exist. It catches the `System.Net.WebException` exception and returns the last value for any other exception.

```
try  
{  
    $wc = new-object System.Net.WebClient  
    $wc.DownloadFile("http://www.contoso.com/MyDoc.doc")  
}  
catch [System.Net.WebException]  
{  
    "Unable to download MyDoc.doc from http://www.contoso.com."  
}  
catch  
{  
    "An error occurred that could not be resolved."  
}
```

Throw

[Throw](#) can be used to generate a terminating error. This mechanism can be useful when defining your own logic in a runbook. If the script meets a criterion that should stop it, it can use the `throw` statement to stop. The following example uses this statement to show a required function parameter.

```
function Get-ContosoFiles  
{  
    param ($path = $(throw "The Path parameter is required."))  
    Get-ChildItem -Path $path\*.txt -recurse  
}
```

Errors

Your runbooks must handle errors. Azure Automation supports two types of PowerShell errors, terminating and non-terminating.

Terminating errors stop runbook execution when they occur. The runbook stops with a job status of Failed.

Non-terminating errors allow a script to continue even after they occur. An example of a non-terminating error is one that occurs when a runbook uses the `Get-ChildItem` cmdlet with a path that doesn't exist. PowerShell sees that the path doesn't exist, throws an error, and continues to the next folder. The error in this case doesn't set the runbook job status to Failed, and the job might even be completed. To force a runbook to stop on a non-terminating error, you can use `ErrorAction Stop` on the cmdlet.

Calling processes

Runbooks that run in Azure sandboxes don't support calling processes, such as executables (.exe files) or subprocesses. The reason for this is that an Azure sandbox is a shared process run in a container that might not be able to access all the underlying APIs. For scenarios requiring third-party software or calls to subprocesses, you should execute a runbook on a [Hybrid Runbook Worker](#).

Device and application characteristics

Runbook jobs in Azure sandboxes can't access any device or application characteristics. The most common API used to query performance metrics on Windows is WMI, with some of the common metrics being memory and CPU usage. However, it doesn't matter what API is used, as jobs running in the cloud can't access the Microsoft implementation of Web-Based Enterprise Management (WBEM). This platform is built on the Common Information Model (CIM), providing the industry standards for defining device and application characteristics.

Webhooks

External services, for example, Azure DevOps Services and GitHub, can start a runbook in Azure Automation. To do this type of startup, the service uses a [webhook](#) via a single HTTP request. Use of a webhook allows runbooks to be started without implementation of a full Azure Automation feature.

Shared resources

To share resources among all runbooks in the cloud, Azure uses a concept called fair share. Using fair share, Azure temporarily unloads or stops any job that has run for more than three hours. Jobs for [PowerShell runbooks](#) and [Python runbooks](#) are stopped and not restarted, and the job status becomes **Stopped**.

For long-running Azure Automation tasks, it's recommended to use a [Hybrid Runbook Worker](#). Hybrid Runbook Workers aren't limited by fair share, and don't have a limitation on how long a runbook can execute. The other job [limits](#) apply to both Azure sandboxes and Hybrid Runbook Workers. While Hybrid Runbook Workers aren't limited by the three-hour fair share limit, you should develop runbooks to run on the workers that support restarts from unexpected local infrastructure issues.

Another option is to optimize a runbook by using child runbooks. For example, your runbook might loop through the same function on several resources, for example, with a database operation on several databases. You can move this function to a [child runbook](#) and have your runbook call it using [Start-AzAutomationRunbook](#). Child runbooks execute in parallel in separate processes.

Using child runbooks decreases the total amount of time for the parent runbook to complete. Your runbook can use the [Get-AzAutomationJob](#) cmdlet to check the job status for a child runbook if it still has more operations after the child completes.

Next steps

- To get started with a PowerShell runbook, see [Tutorial: Create a PowerShell runbook](#).
- To work with runbooks, see [Manage runbooks in Azure Automation](#).
- For details of PowerShell, see [PowerShell Docs](#).
- For a PowerShell cmdlet reference, see [Az.Automation](#).

Hybrid Runbook Worker overview

7/23/2021 • 8 minutes to read • [Edit Online](#)

Runbooks in Azure Automation might not have access to resources in other clouds or in your on-premises environment because they run on the Azure cloud platform. You can use the Hybrid Runbook Worker feature of Azure Automation to run runbooks directly on the machine that's hosting the role and against resources in the environment to manage those local resources. Runbooks are stored and managed in Azure Automation and then delivered to one or more assigned machines.

Runbook Worker types

There are two types of Runbook Workers - system and user. The following table describes the difference between them.

TYPE	DESCRIPTION
System	Supports a set of hidden runbooks used by the Update Management feature that are designed to install user-specified updates on Windows and Linux machines. This type of Hybrid Runbook Worker is not a member of a Hybrid Runbook Worker group, and therefore doesn't run runbooks that target a Runbook Worker group.
User	Supports user-defined runbooks intended to run directly on the Windows and Linux machine that are members of one or more Runbook Worker groups.

A Hybrid Runbook Worker can run on either the Windows or the Linux operating system, and this role relies on the [Log Analytics agent](#) reporting to an Azure Monitor [Log Analytics workspace](#). The workspace is not only to monitor the machine for the supported operating system, but also to download the components required to install the Hybrid Runbook Worker.

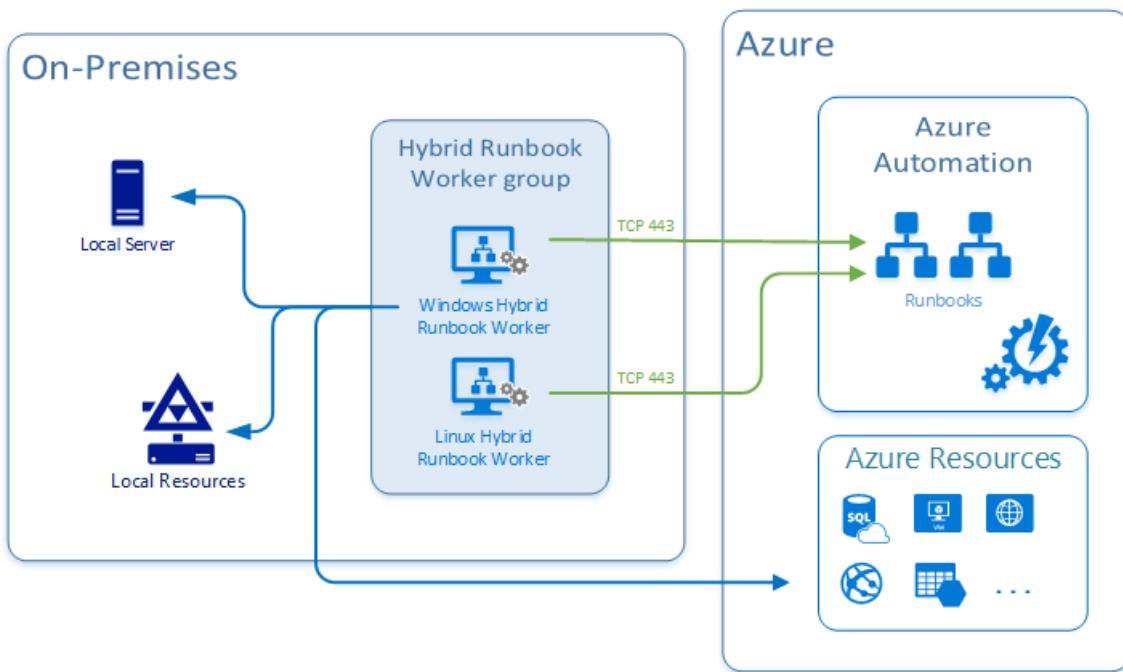
When Azure Automation [Update Management](#) is enabled, any machine connected to your Log Analytics workspace is automatically configured as a system Hybrid Runbook Worker. To configure it as a user Windows Hybrid Runbook Worker, see [Deploy a Windows Hybrid Runbook Worker](#) and for Linux, see [Deploy a Linux Hybrid Runbook Worker](#).

Runbook Worker limits

The following table shows the maximum number of system and user hybrid runbook workers in an Automation account. If you have more than 4,000 machines to manage, we recommend creating another Automation account.

WORKER TYPE	MAXIMUM NUMBER SUPPORTED PER AUTOMATION ACCOUNT.
System	4000
User	4000

How does it work?



Each user Hybrid Runbook Worker is a member of a Hybrid Runbook Worker group that you specify when you install the worker. A group can include a single worker, but you can include multiple workers in a group for high availability. Each machine can host one Hybrid Runbook Worker reporting to one Automation account; you cannot register the hybrid worker across multiple Automation accounts. A hybrid worker can only listen for jobs from a single Automation account. For machines hosting the system Hybrid Runbook worker managed by Update Management, they can be added to a Hybrid Runbook Worker group. But you must use the same Automation account for both Update Management and the Hybrid Runbook Worker group membership.

When you start a runbook on a user Hybrid Runbook Worker, you specify the group that it runs on. Each worker in the group polls Azure Automation to see if any jobs are available. If a job is available, the first worker to get the job takes it. The processing time of the jobs queue depends on the hybrid worker hardware profile and load. You can't specify a particular worker. Hybrid worker works on a polling mechanism (every 30 secs) and follows an order of first-come, first-serve. Depending on when a job was pushed, whichever hybrid worker pings the Automation service picks up the job. A single hybrid worker can generally pick up four jobs per ping (that is, every 30 seconds). If your rate of pushing jobs is higher than four per 30 seconds, then there is a high possibility another hybrid worker in the Hybrid Runbook Worker group picked up the job.

A Hybrid Runbook Worker doesn't have many of the [Azure sandbox](#) resource limits on disk space, memory, or network sockets. The limits on a hybrid worker are only related to the worker's own resources, and they aren't constrained by the [fair share](#) time limit that Azure sandboxes have.

To control the distribution of runbooks on Hybrid Runbook Workers and when or how the jobs are triggered, you can register the hybrid worker against different Hybrid Runbook Worker groups within your Automation account. Target the jobs against the specific group or groups in order to support your execution arrangement.

Hybrid Runbook Worker installation

The process to install a user Hybrid Runbook Worker depends on the operating system. The table below defines the deployment types.

OPERATING SYSTEM	DEPLOYMENT TYPES
Windows	Automated Manual
Linux	Manual

The recommended installation method for a Windows machine is to use an Azure Automation runbook to completely automate the process of configuring it. If that isn't feasible, you can follow a step-by-step procedure to manually install and configure the role. For Linux machines, you run a Python script to install the agent on the machine.

Network planning

Check [Azure Automation Network Configuration](#) for detailed information on the ports, URLs, and other networking details required for the Hybrid Runbook Worker.

Proxy server use

If you use a proxy server for communication between Azure Automation and machines running the Log Analytics agent, ensure that the appropriate resources are accessible. The timeout for requests from the Hybrid Runbook Worker and Automation services is 30 seconds. After three attempts, a request fails.

Firewall use

If you use a firewall to restrict access to the Internet, you must configure the firewall to permit access. If using the Log Analytics gateway as a proxy, ensure that it is configured for Hybrid Runbook Workers. See [Configure the Log Analytics gateway for Automation Hybrid Runbook Workers](#).

Service tags

Azure Automation supports Azure virtual network service tags, starting with the service tag **GuestAndHybridManagement**. You can use service tags to define network access controls on [network security groups](#) or [Azure Firewall](#). Service tags can be used in place of specific IP addresses when you create security rules. By specifying the service tag name **GuestAndHybridManagement** in the appropriate source or destination field of a rule, you can allow or deny the traffic for the Automation service. This service tag does not support allowing more granular control by restricting IP ranges to a specific region.

The service tag for the Azure Automation service only provides IPs used for the following scenarios:

- Trigger webhooks from within your virtual network
- Allow Hybrid Runbook Workers or State Configuration agents on your VNet to communicate with the Automation service

NOTE

The service tag **GuestAndHybridManagement** currently doesn't support runbook job execution in an Azure sandbox, only directly on a Hybrid Runbook Worker.

Support for Impact Level 5 (IL5)

Azure Automation Hybrid Runbook Worker can be used in Azure Government to support Impact Level 5 workloads in either of the following two configurations:

- [Isolated virtual machine](#). When deployed, they consume the entire physical host for that machine providing the necessary level of isolation required to support IL5 workloads.
- [Azure Dedicated Hosts](#), which provides physical servers that are able to host one or more virtual machines, dedicated to one Azure subscription.

NOTE

Compute isolation through the Hybrid Runbook Worker role is available for Azure Commercial and US Government clouds.

Update Management addresses for Hybrid Runbook Worker

In addition to the standard addresses and ports required for the Hybrid Runbook Worker, Update Management has other network configuration requirements described under the [network planning](#) section.

Azure Automation State Configuration on a Hybrid Runbook Worker

You can run [Azure Automation State Configuration](#) on a Hybrid Runbook Worker. To manage the configuration of servers that support the Hybrid Runbook Worker, you must add the servers as DSC nodes. See [Enable machines for management by Azure Automation State Configuration](#).

Runbooks on a Hybrid Runbook Worker

You might have runbooks that manage resources on the local machine or run against resources in the local environment where a user Hybrid Runbook Worker is deployed. In this case, you can choose to run your runbooks on the hybrid worker instead of in an Automation account. Runbooks run on a Hybrid Runbook Worker are identical in structure to those that you run in the Automation account. See [Run runbooks on a Hybrid Runbook Worker](#).

Hybrid Runbook Worker jobs

Hybrid Runbook Worker jobs run under the local **System** account on Windows or the [nxautomation account](#) on Linux. Azure Automation handles jobs on Hybrid Runbook Workers differently from jobs run in Azure sandboxes. See [Runbook execution environment](#).

If the Hybrid Runbook Worker host machine reboots, any running runbook job restarts from the beginning, or from the last checkpoint for PowerShell Workflow runbooks. After a runbook job is restarted more than three times, it is suspended.

Runbook permissions for a Hybrid Runbook Worker

Since they access non-Azure resources, runbooks running on a user Hybrid Runbook Worker can't use the authentication mechanism typically used by runbooks authenticating to Azure resources. A runbook either provides its own authentication to local resources, or configures authentication using [managed identities for Azure resources](#). You can also specify a Run As account to provide a user context for all runbooks.

View system Hybrid Runbook Workers

After the Update Management feature is enabled on Windows or Linux machines, you can inventory the list of system Hybrid Runbook Workers group in the Azure portal. You can view up to 2,000 workers in the portal by selecting the tab **System hybrid workers group** from the option **Hybrid workers group** from the left-hand pane for the selected Automation account.

Group name	Number of workers	Last registration time	Last seen time
AppBE00.NA.contosohotels.com_a8b5...	1	9/18/2019, 6:56 AM	27 minutes ago
AppBE01.NA.contosohotels.com_d231...	1	9/18/2019, 6:56 AM	1 minute ago
AppFE000000Q_36eb2c84-c065-412f...	1	5/11/2020, 1:44 PM	1 minute ago
AppFE000000R_b3097b99-1262-4565...	1	5/11/2020, 1:47 PM	1 minute ago
AppFE00002TI_3597694b-87d2-4ea9...	1	8/13/2020, 10:18 PM	8/13/2020, 11:49 PM
AppFE00002TJ_d5376514-3fc8-497f-b...	1	8/13/2020, 10:22 PM	8/13/2020, 11:53 PM
AppFE00002TK_c844bb12-72cd-49ec...	1	8/13/2020, 10:24 PM	8/13/2020, 10:25 PM
AppFE00002TL_a5d6a52c-c0e4-46ac...	1	8/13/2020, 10:24 PM	8/13/2020, 10:24 PM
AppFE00002TM_7c782e44-dd8c-4352...	1	8/14/2020, 1:23 AM	8/14/2020, 2:54 AM
AppFE00002TN_f218cfae-5b0a-40d7...	1	8/14/2020, 1:23 AM	8/14/2020, 1:23 AM
AppFE00002TO_cb45f044-a5eb-4e58...	1	8/14/2020, 1:24 AM	8/14/2020, 1:25 AM
AppFE00002TP_e52fb59e-1ac7-4ac5-b...	1	8/14/2020, 1:24 AM	8/14/2020, 2:55 AM
AppFE00002TQ_c4274b6e-12bc-418c-	1	8/14/2020, 4:21 AM	8/14/2020, 4:21 AM

If you have more than 2,000 hybrid workers, to get a list of all of them, you can run the following PowerShell script:

```
"Get-AzSubscription -SubscriptionName "<subscriptionName>" | Set-AzContext
$workersList = (Get-AzAutomationHybridWorkerGroup -ResourceGroupName "<resourceGroupName>" -
AutomationAccountName "<automationAccountName>").Runbookworker
$workersList | export-csv -Path "<Path>\output.csv" -NoClobber -NoTypeInformation"
```

Next steps

- To learn how to configure your runbooks to automate processes in your on-premises datacenter or other cloud environment, see [Run runbooks on a Hybrid Runbook Worker](#).
- To learn how to troubleshoot your Hybrid Runbook Workers, see [Troubleshoot Hybrid Runbook Worker issues](#).

Azure Automation runbook types

9/10/2021 • 5 minutes to read • [Edit Online](#)

The Azure Automation Process Automation feature supports several types of runbooks, as defined in the following table. To learn about the process automation environment, see [Runbook execution in Azure Automation](#).

TYPE	DESCRIPTION
Graphical	Graphical runbook based on Windows PowerShell and created and edited completely in the graphical editor in Azure portal.
Graphical PowerShell Workflow	Graphical runbook based on Windows PowerShell Workflow and created and edited completely in the graphical editor in Azure portal.
PowerShell	Textual runbook based on Windows PowerShell scripting.
PowerShell Workflow	Textual runbook based on Windows PowerShell Workflow scripting.
Python	Textual runbook based on Python scripting.

Take into account the following considerations when determining which type to use for a particular runbook.

- You can't convert runbooks from graphical to text type, or the other way around.
- There are limitations when using runbooks of different types as child runbooks. For more information, see [Child runbooks in Azure Automation](#).

Graphical runbooks

You can create and edit graphical and graphical PowerShell Workflow runbooks using the graphical editor in the Azure portal. However, you can't create or edit this type of runbook with another tool. Main features of graphical runbooks:

- Exported to files in your Automation account and then imported into another Automation account.
- Generate PowerShell code.
- Converted to or from graphical PowerShell Workflow runbooks during import.

Advantages

- Use visual insert-link-configure authoring model.
- Focus on how data flows through the process.
- Visually represent management processes.
- Include other runbooks as child runbooks to create high-level workflows.
- Encourage modular programming.

Limitations

- Can't create or edit outside the Azure portal.
- Might require a code activity containing PowerShell code to execute complex logic.

- Can't convert to one of the [text formats](#), nor can you convert a text runbook to graphical format.
- Can't view or directly edit PowerShell code that the graphical workflow creates. You can view the code you create in any code activities.
- Can't run runbooks on a Linux Hybrid Runbook Worker. See [Automate resources in your datacenter or cloud by using Hybrid Runbook Worker](#).

PowerShell runbooks

PowerShell runbooks are based on Windows PowerShell. You directly edit the code of the runbook using the text editor in the Azure portal. You can also use any offline text editor and [import the runbook](#) into Azure Automation.

Advantages

- Implement all complex logic with PowerShell code without the other complexities of PowerShell Workflow.
- Start faster than PowerShell Workflow runbooks, since they don't need to be compiled before running.
- Run in Azure and on Hybrid Runbook Workers for both Windows and Linux.

Limitations

- You must be familiar with PowerShell scripting.
- Runbooks can't use [parallel processing](#) to execute multiple actions in parallel.
- Runbooks can't use [checkpoints](#) to resume runbook if there's an error.
- You can include only PowerShell Workflow runbooks and graphical runbooks as child runbooks by using the [Start-AzAutomationRunbook](#) cmdlet, which creates a new job.
- Runbooks can't use the PowerShell [#Requires](#) statement, it is not supported in Azure sandbox or on Hybrid Runbook Workers and will cause the job to fail.

Known issues

The following are current known issues with PowerShell runbooks:

- PowerShell runbooks can't retrieve an unencrypted [variable asset](#) with a null value.
- PowerShell runbooks can't retrieve a variable asset with `*~*` in the name.
- A [Get-Process](#) operation in a loop in a PowerShell runbook can crash after about 80 iterations.
- A PowerShell runbook can fail if it tries to write a large amount of data to the output stream at once. You can typically work around this issue by having the runbook output just the information needed to work with large objects. For example, instead of using `Get-Process` with no limitations, you can have the cmdlet output just the required parameters as in `Get-Process | Select ProcessName, CPU`.

PowerShell Workflow runbooks

PowerShell Workflow runbooks are text runbooks based on [Windows PowerShell Workflow](#). You directly edit the code of the runbook using the text editor in the Azure portal. You can also use any offline text editor and [import the runbook](#) into Azure Automation.

Advantages

- Implement all complex logic with PowerShell Workflow code.
- Use [checkpoints](#) to resume operation if there's an error.
- Use [parallel processing](#) to do multiple actions in parallel.
- Can include other graphical runbooks and PowerShell Workflow runbooks as child runbooks to create high-level workflows.

Limitations

- You must be familiar with PowerShell Workflow.

- Runbooks must deal with the additional complexity of PowerShell Workflow, such as [deserialized objects](#).
- Runbooks take longer to start than PowerShell runbooks since they must be compiled before running.
- You can only include PowerShell runbooks as child runbooks by using the `Start-AzAutomationRunbook` cmdlet.
- Runbooks can't run on a Linux Hybrid Runbook Worker.

Python runbooks

Python runbooks compile under Python 2 and Python 3. Python 3 runbooks are currently in preview. You can directly edit the code of the runbook using the text editor in the Azure portal. You can also use an offline text editor and [import the runbook](#) into Azure Automation.

Python 3 runbooks are supported in the following Azure global infrastructures:

- Azure global
- Azure Government

Advantages

- Use the robust Python libraries.
- Can run in Azure or on Hybrid Runbook Workers.
- For Python 2, Windows Hybrid Runbook Workers are supported with [python 2.7](#) installed.
- For Python 3 Cloud Jobs, Python 3.8 version is supported. Scripts and packages from any 3.x version might work if the code is compatible across different versions.
- For Python 3 Hybrid jobs on Windows machines, you can choose to install any 3.x version you may want to use.
- For Python 3 Hybrid jobs on Linux machines, we depend on the Python 3 version installed on the machine to run DSC OMSConfig and the Linux Hybrid Worker. We recommend installing 3.6 on Linux machines. However, different versions should also work if there are no breaking changes in method signatures or contracts between versions of Python 3.

Limitations

- You must be familiar with Python scripting.
- To use third-party libraries, you must [import the packages](#) into the Automation account.
- Using `Start-AutomationRunbook` cmdlet in PowerShell/PowerShell Workflow to start a Python 3 runbook (preview) doesn't work. You can use `Start-AzAutomationRunbook` cmdlet from Az.Automation module or `Start-AzureRmAutomationRunbook` cmdlet from AzureRm.Automation module to work around this limitation.
- Azure Automation doesn't support `sys.stderr`.

Known issues

For cloud jobs, Python 3 jobs sometimes fail with an exception message `invalid interpreter executable path`. You might see this exception if the job is delayed, starting more than 10 minutes, or using `Start-AutomationRunbook` to start Python 3 runbooks. If the job is delayed, restarting the runbook should be sufficient. Hybrid jobs should work without any issue if using the following steps:

1. Create a new environment variable called `PYTHON_3_PATH` and specify the installation folder. For example, if the installation folder is `C:\Python3`, then this path needs to be added to the variable.
2. Restart the machine after setting the environment variable.

Next steps

- To learn about PowerShell runbooks, see [Tutorial: Create a PowerShell runbook](#).
- To learn about PowerShell Workflow runbooks, see [Tutorial: Create a PowerShell Workflow runbook](#).

- To learn about graphical runbooks, see [Tutorial: Create a graphical runbook](#).
- To learn about Python runbooks, see [Tutorial: Create a Python runbook](#).

Azure Automation network configuration details

7/19/2021 • 2 minutes to read • [Edit Online](#)

This page provides networking details that are required for [Hybrid Runbook Worker and State Configuration](#), and for [Update Management and Change Tracking and Inventory](#).

Hybrid Runbook Worker and State Configuration

The following port and URLs are required for the Hybrid Runbook Worker, and for [Automation State Configuration](#) to communicate with Azure Automation.

- Port: Only 443 required for outbound internet access
- Global URL: `*.azure-automation.net`
- Global URL of US Gov Virginia: `*.azure-automation.us`
- Agent service: `https://<workspaceId>.agentsvc.azure-automation.net`

Network planning for Hybrid Runbook Worker

For either a system or user Hybrid Runbook Worker to connect to and register with Azure Automation, it must have access to the port number and URLs described in this section. The worker must also have access to the [ports and URLs required for the Log Analytics agent](#) to connect to the Azure Monitor Log Analytics workspace.

If you have an Automation account that's defined for a specific region, you can restrict Hybrid Runbook Worker communication to that regional datacenter. Review the [DNS records used by Azure Automation](#) for the required DNS records.

Configuration of private networks for State Configuration

If your nodes are located in a private network, the port and URLs defined above are required. These resources provide network connectivity for the managed node and allow DSC to communicate with Azure Automation.

If you are using DSC resources that communicate between nodes, such as the [WaitFor* resources](#), you also need to allow traffic between nodes. See the documentation for each DSC resource to understand these network requirements.

To understand client requirements for TLS 1.2, see [TLS 1.2 for Azure Automation](#).

Update Management and Change Tracking and Inventory

The addresses in this table are required both for Update Management and for Change Tracking and Inventory. The paragraph following the table also applies to both.

Communication to these addresses uses **port 443**.

AZURE PUBLIC	AZURE GOVERNMENT
<code>*.ods.opinsights.azure.com</code>	<code>*.ods.opinsights.azure.us</code>
<code>*.oms.opinsights.azure.com</code>	<code>*.oms.opinsights.azure.us</code>
<code>*.blob.core.windows.net</code>	<code>*.blob.core.usgovcloudapi.net</code>
<code>*.azure-automation.net</code>	<code>*.azure-automation.us</code>

When you create network group security rules or configure Azure Firewall to allow traffic to the Automation service and the Log Analytics workspace, use the [service tags GuestAndHybridManagement](#) and [AzureMonitor](#). This simplifies the ongoing management of your network security rules. To connect to the Automation service from your Azure VMs securely and privately, review [Use Azure Private Link](#). To obtain the current service tag and range information to include as part of your on-premises firewall configurations, see [downloadable JSON files](#).

Next steps

- Learn about [Automation Update Management overview](#).
- Learn about [Hybrid Runbook Worker](#).
- Learn about [Change Tracking and Inventory](#).
- Learn about [Automation State Configuration](#).

Azure Policy Regulatory Compliance controls for Azure Automation

9/13/2021 • 5 minutes to read • [Edit Online](#)

Regulatory Compliance in Azure Policy provides Microsoft created and managed initiative definitions, known as **built-ins**, for the **compliance domains** and **security controls** related to different compliance standards. This page lists the **compliance domains** and **security controls** for Azure Automation. You can assign the built-ins for a **security control** individually to help make your Azure resources compliant with the specific standard.

The title of each built-in policy definition links to the policy definition in the Azure portal. Use the link in the **Policy Version** column to view the source on the [Azure Policy GitHub repo](#).

IMPORTANT

Each control below is associated with one or more Azure Policy definitions. These policies may help you [assess compliance](#) with the control; however, there often is not a one-to-one or complete match between a control and one or more policies. As such, **Compliant** in Azure Policy refers only to the policies themselves; this doesn't ensure you're fully compliant with all requirements of a control. In addition, the compliance standard includes controls that aren't addressed by any Azure Policy definitions at this time. Therefore, compliance in Azure Policy is only a partial view of your overall compliance status. The associations between controls and Azure Policy Regulatory Compliance definitions for these compliance standards may change over time.

Azure Security Benchmark

The [Azure Security Benchmark](#) provides recommendations on how you can secure your cloud solutions on Azure. To see how this service completely maps to the Azure Security Benchmark, see the [Azure Security Benchmark mapping files](#).

To review how the available Azure Policy built-ins for all Azure services map to this compliance standard, see [Azure Policy Regulatory Compliance - Azure Security Benchmark](#).

DOMAIN	CONTROL ID	CONTROL TITLE	POLICY (AZURE PORTAL)	POLICY VERSION (GITHUB)
Data Protection	DP-5	Encrypt sensitive data at rest	Automation account variables should be encrypted	1.1.0

Azure Security Benchmark v1

The [Azure Security Benchmark](#) provides recommendations on how you can secure your cloud solutions on Azure. To see how this service completely maps to the Azure Security Benchmark, see the [Azure Security Benchmark mapping files](#).

To review how the available Azure Policy built-ins for all Azure services map to this compliance standard, see [Azure Policy Regulatory Compliance - Azure Security Benchmark](#).

DOMAIN	CONTROL ID	CONTROL TITLE	POLICY (AZURE PORTAL)	POLICY VERSION (GITHUB)
Data Protection	4.8	Encrypt sensitive information at rest	Automation account variables should be encrypted	1.1.0

CMMC Level 3

To review how the available Azure Policy built-ins for all Azure services map to this compliance standard, see [Azure Policy Regulatory Compliance - CMMC Level 3](#). For more information about this compliance standard, see [Cybersecurity Maturity Model Certification \(CMMC\)](#).

DOMAIN	CONTROL ID	CONTROL TITLE	POLICY (AZURE PORTAL)	POLICY VERSION (GITHUB)
System and Communications Protection	SC.3.177	Employ FIPS-validated cryptography when used to protect the confidentiality of CUI.	Automation account variables should be encrypted	1.1.0
System and Communications Protection	SC.3.191	Protect the confidentiality of CUI at rest.	Automation account variables should be encrypted	1.1.0

FedRAMP High

To review how the available Azure Policy built-ins for all Azure services map to this compliance standard, see [Azure Policy Regulatory Compliance - FedRAMP High](#). For more information about this compliance standard, see [FedRAMP High](#).

DOMAIN	CONTROL ID	CONTROL TITLE	POLICY (AZURE PORTAL)	POLICY VERSION (GITHUB)
System and Communications Protection	SC-12	Cryptographic Key Establishment and Management	Azure Automation accounts should use customer-managed keys to encrypt data at rest	1.0.0
System and Communications Protection	SC-28	Protection of Information at Rest	Automation account variables should be encrypted	1.1.0
System and Communications Protection	SC-28 (1)	Cryptographic Protection	Automation account variables should be encrypted	1.1.0

FedRAMP Moderate

To review how the available Azure Policy built-ins for all Azure services map to this compliance standard, see [Azure Policy Regulatory Compliance - FedRAMP Moderate](#). For more information about this compliance standard, see [FedRAMP Moderate](#).

DOMAIN	CONTROL ID	CONTROL TITLE	POLICY (AZURE PORTAL)	POLICY VERSION (GITHUB)
System and Communications Protection	SC-12	Cryptographic Key Establishment and Management	Azure Automation accounts should use customer-managed keys to encrypt data at rest	1.0.0
System and Communications Protection	SC-28	Protection of Information at Rest	Automation account variables should be encrypted	1.1.0
System and Communications Protection	SC-28 (1)	Cryptographic Protection	Automation account variables should be encrypted	1.1.0

ISO 27001:2013

To review how the available Azure Policy built-ins for all Azure services map to this compliance standard, see [Azure Policy Regulatory Compliance - ISO 27001:2013](#). For more information about this compliance standard, see [ISO 27001:2013](#).

DOMAIN	CONTROL ID	CONTROL TITLE	POLICY (AZURE PORTAL)	POLICY VERSION (GITHUB)
Cryptography	10.1.1	Policy on the use of cryptographic controls	Automation account variables should be encrypted	1.1.0

NIST SP 800-53 Rev. 4

To review how the available Azure Policy built-ins for all Azure services map to this compliance standard, see [Azure Policy Regulatory Compliance - NIST SP 800-53 Rev. 4](#). For more information about this compliance standard, see [NIST SP 800-53 Rev. 4](#).

DOMAIN	CONTROL ID	CONTROL TITLE	POLICY (AZURE PORTAL)	POLICY VERSION (GITHUB)
System and Communications Protection	SC-12	Cryptographic Key Establishment and Management	Azure Automation accounts should use customer-managed keys to encrypt data at rest	1.0.0
System and Communications Protection	SC-28	Protection of Information at Rest	Automation account variables should be encrypted	1.1.0
System and Communications Protection	SC-28 (1)	Cryptographic Protection	Automation account variables should be encrypted	1.1.0

NIST SP 800-53 Rev. 5

To review how the available Azure Policy built-ins for all Azure services map to this compliance standard, see

Azure Policy Regulatory Compliance - NIST SP 800-53 Rev. 5. For more information about this compliance standard, see [NIST SP 800-53 Rev. 5](#).

DOMAIN	CONTROL ID	CONTROL TITLE	POLICY (AZURE PORTAL)	POLICY VERSION (GITHUB)
System and Communications Protection	SC-12	Cryptographic Key Establishment and Management	Azure Automation accounts should use customer-managed keys to encrypt data at rest	1.0.0
System and Communications Protection	SC-28	Protection of Information at Rest	Automation account variables should be encrypted	1.1.0
System and Communications Protection	SC-28 (1)	Cryptographic Protection	Automation account variables should be encrypted	1.1.0

UK OFFICIAL and UK NHS

To review how the available Azure Policy built-ins for all Azure services map to this compliance standard, see [Azure Policy Regulatory Compliance - UK OFFICIAL and UK NHS](#). For more information about this compliance standard, see [UK OFFICIAL](#).

DOMAIN	CONTROL ID	CONTROL TITLE	POLICY (AZURE PORTAL)	POLICY VERSION (GITHUB)
Asset protection and resilience	2.3	Data at rest protection	Automation account variables should be encrypted	1.1.0

Next steps

- Learn more about [Azure Policy Regulatory Compliance](#).
- See the built-ins on the [Azure Policy GitHub repo](#).

Encryption of secure assets in Azure Automation

8/3/2021 • 6 minutes to read • [Edit Online](#)

Secure assets in Azure Automation include credentials, certificates, connections, and encrypted variables. These assets are protected in Azure Automation using multiple levels of encryption. Based on the top-level key used for the encryption, there are two models for encryption:

- Using Microsoft-managed keys
- Using keys that you manage

Microsoft-managed Keys

By default, your Azure Automation account uses Microsoft-managed keys.

Each secure asset is encrypted and stored in Azure Automation using a unique key (Data Encryption key) that is generated for each automation account. These keys themselves are encrypted and stored in Azure Automation using yet another unique key that is generated for each account called an Account Encryption Key (AEK). These account encryption keys encrypted and stored in Azure Automation using Microsoft-managed Keys.

Keys that you manage with Key Vault

You can manage encryption of secure assets for your Automation account with your own keys. When you specify a customer-managed key at the level of the Automation account, that key is used to protect and control access to the account encryption key for the Automation account. This in turn is used to encrypt and decrypt all the secure assets. Customer-managed keys offer greater flexibility to create, rotate, disable, and revoke access controls. You can also audit the encryption keys used to protect your secure assets.

Use Azure Key Vault to store customer-managed keys. You can either create your own keys and store them in a key vault, or you can use the Azure Key Vault APIs to generate keys.

Enabling the Azure Firewall on [Azure Key Vault](#) blocks access from Azure Automation runbooks for that service. Access will be blocked even when the firewall exception to allow trusted Microsoft services is enabled, as Automation is not a part of the trusted services list. With an enabled firewall, access can only be made by using a Hybrid Runbook Worker and a [virtual network service endpoint](#).

For more information about Azure Key Vault, see [What is Azure Key Vault?](#)

Use of customer-managed keys for an Automation account

When you use encryption with customer-managed keys for an Automation account, Azure Automation wraps the account encryption key with the customer-managed key in the associated key vault. Enabling customer-managed keys doesn't impact performance, and the account is encrypted with the new key immediately, without any delay.

A new Automation account is always encrypted using Microsoft-managed keys. It's not possible to enable customer-managed keys at the time that the account is created. Customer-managed keys are stored in Azure Key Vault, and the key vault must be provisioned with access policies that grant key permissions to the managed identity that is associated with the Automation account. The managed identity is available only after the storage account is created.

When you modify the key being used for Azure Automation secure asset encryption, by enabling or disabling customer-managed keys, updating the key version, or specifying a different key, the encryption of the account

encryption key changes but the secure assets in your Azure Automation account don't need to be re-encrypted.

NOTE

To enable customer-managed key using Azure Automation REST API calls, you need to use api version 2020-01-13-preview.

Prerequisites for using customer-managed keys in Azure Automation

Before enabling customer-managed keys for an Automation account, you must ensure the following prerequisites are met:

- An [Azure Key Vault](#) with the **Soft Delete** and **Do Not Purge** properties enabled. These properties are required to allow for recovery of keys if there's accidental deletion.
- Only RSA keys are supported with Azure Automation encryption. For more information about keys, see [About Azure Key Vault keys, secrets, and certificates](#).
- The Automation account and the key vault can be in different subscriptions but need to be in the same Azure Active Directory tenant.
- When using PowerShell, verify the [Azure Az PowerShell module](#) is installed. To install or upgrade, see [How to install the Azure Az PowerShell module](#).

Generate and assign a new system-assigned identity for an Automation account

To use customer-managed keys with an Automation account, your Automation account needs to authenticate against the key vault storing customer-managed keys. Azure Automation uses system assigned managed identities to authenticate the account with Azure Key Vault. For more information about managed identities, see [What are managed identities for Azure resources?](#)

Using PowerShell

Use PowerShell cmdlet [Set-AzAutomationAccount](#) to modify an existing Azure Automation account. The `-AssignSystemIdentity` parameter generates and assigns a new system-assigned identity for the Automation account to use with other services like Azure Key Vault. For more information, see [What are managed identities for Azure resources?](#) and [About Azure Key Vault](#). Execute the following code:

```
# Revise variables with your actual values.  
$resourceGroup = "ResourceGroupName"  
$automationAccount = "AutomationAccountName"  
$vaultName = "KeyVaultName"  
$keyName = "KeyName"  
  
Set-AzAutomationAccount `  
    -ResourceGroupName $resourceGroup `  
    -Name $automationAccount `  
    -AssignSystemIdentity
```

The output should look similar to the following:

```

SubscriptionId      : ContosoID
ResourceGroupName   : Contosolab
AutomationAccountName : ContosoAutomation
Location           : westus
State               : Ok
Plan                : Basic
CreationTime        : 6/24/2021 5:02:33 PM +00:00
LastModifiedTime    : 6/25/2021 7:32:49 PM +00:00
LastModifiedBy      :
Tags                : []
Identity            : Microsoft.Azure.Management.Automation.Models.Identity
Encryption          : Microsoft.Azure.Management.Automation.Models.EncryptionProperties
PublicNetworkAccess  :

```

Obtain the `PrincipalId` for later use. Execute the following code:

```

$principalID = (Get-AzAutomationAccount ` 
    -ResourceGroupName $resourceGroup ` 
    -Name $automationAccount).Identity.PrincipalID

$principalID

```

Using REST

Configure a system-assigned managed identity to the Automation account using the following REST API call:

```

PATCH https://management.azure.com/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/resource-group-name/providers/Microsoft.Automation/automationAccounts/automation-account-name?api-version=2020-01-13-preview

```

Request body:

```
{
  "identity": {
    {
      "type": "SystemAssigned"
    }
  }
}
```

System-assigned identity for the Automation account is returned in a response similar to the following:

```
{
  "name": "automation-account-name",
  "id": "/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/resource-group-name/providers/Microsoft.Automation/automationAccounts/automation-account-name",
  ..
  "identity": {
    "type": "SystemAssigned",
    "principalId": "aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaa",
    "tenantId": "bbbbbbbb-bbbb-bbbb-bbbb-bbbbbbbbbb"
  },
  ..
}
```

Configuration of the Key Vault access policy

Once a system assigned managed identity is assigned to the Automation account, you configure access to the key vault storing customer-managed keys. Azure Automation requires the `Get`, `Recover`, `WrapKey`, and `UnwrapKey` operation permissions for the identity to access the customer-managed keys.

Using PowerShell

Use PowerShell cmdlet [Set-AzKeyVaultAccessPolicy](#) to grant the necessary permissions. Then use [Add-AzKeyVaultKey](#) to create a key in the key vault. Execute the following code:

```
Set-AzKeyVaultAccessPolicy ` 
    -VaultName $vaultName ` 
    -ObjectId $principalID ` 
    -PermissionsToKeys Get, Recover, UnwrapKey, WrapKey

Add-AzKeyVaultKey ` 
    -VaultName $vaultName ` 
    -Name $keyName ` 
    -Destination 'Software'
```

The output should look similar to the following:

```
Vault/HSM Name : contosovault
Name          : contosokey
Key Type      : RSA
Key Size      : 2048
Version       : 136065337ca7446
Id            : https://contosovault.azure.net:443/keys/contosoKey/12345

Enabled        : True
Expires        :
Not Before    :
Created        : 6/25/2021 8:50:06 PM
Updated        : 6/25/2021 8:50:06 PM
Recovery Level : Recoverable
Tags          :
```

Using REST

The access policy can be set using the following REST API call:

```
PUT https://management.azure.com/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/sample-group/providers/Microsoft.KeyVault/vaults/sample-vault/accessPolicies/add?api-version=2018-02-14
```

Request body:

```
{
  "properties": {
    "accessPolicies": [
      {
        "tenantId": "bbbbbbbb-bbbb-bbbb-bbbb-bbbbbbbbbb",
        "objectId": "aaaaaaaa-aaaa-aaaa-aaa-aaaaaaaaaa",
        "permissions": {
          "keys": [
            "get",
            "recover",
            "wrapKey",
            "unwrapKey"
          ],
          "secrets": [],
          "certificates": []
        }
      }
    ]
  }
}
```

NOTE

The **tenantId** and **objectId** fields must be provided with values of **identity.tenantId** and **identity.principalId** respectively from the response of managed identity for the Automation account.

Reconfigure Automation account to use customer-managed key

If you want to switch your Automation account from Microsoft-managed keys to customer-managed keys, you can perform this change using Azure PowerShell or with an Azure Resource Manager template.

Using PowerShell

Use PowerShell cmdlet [Set-AzAutomationAccount](#) to reconfigure the Automation account to use customer-managed keys.

```
$vaultURI = (Get-AzKeyVault -VaultName $vaultName).VaultUri  
$keyVersion = (Get-AzKeyVaultKey -VaultName $vaultName -KeyName $keyName).Version  
  
Set-AzAutomationAccount `  
    -ResourceGroupName $resourceGroup `  
    -Name $automationAccount `  
    -AssignSystemIdentity `  
    -KeyName $keyName `  
    -KeyVaultUri $vaultURI `  
    -KeyVersion $keyVersion `  
    -KeyVaultEncryption
```

You can verify the change by running the following command:

```
(Get-AzAutomationAccount `  
    -ResourceGroupName $resourceGroup `  
    -Name $automationAccount).Encryption `  
| ConvertTo-Json
```

The output should look similar to the following:

```
{  
    "KeyVaultProperties": {  
        "KeyVaultUri": "https://contosovault.vault.azure.net/",  
        "KeyName": "contosoKey",  
        "KeyVersion": "6df3748509281153"  
    },  
    "KeySource": 1,  
    "Identity": {  
        "UserAssignedIdentity": null  
    }  
}
```

Using REST

Use the following REST API call:

```
PATCH https://management.azure.com/subscriptions/00000000-0000-0000-0000-  
000000000000/resourceGroups/resource-group-  
name/providers/Microsoft.Automation/automationAccounts/automation-account-name?api-version=2020-01-13-  
preview
```

Request body:

```
{  
  "identity": {  
    "type": "SystemAssigned"  
  },  
  "properties": {  
    "encryption": {  
      "keySource": "Microsoft.Keyvault",  
      "keyvaultProperties": {  
        "keyName": "sample-vault-key",  
        "keyvaultUri": "https://sample-vault-key12.vault.azure.net",  
        "keyVersion": "7c73556c521340209371eaf623cc099d"  
      }  
    }  
  }  
}
```

Sample response

```
{  
  "name": "automation-account-name",  
  "id": "/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/resource-group-  
  name/providers/Microsoft.Automation/automationAccounts/automation-account-name",  
  ..  
  "properties": {  
    ..  
    "encryption": {  
      "keyvaultProperties": {  
        "keyName": "sample-vault-key",  
        "keyvaultUri": "https://sample-vault-key12.vault.azure.net",  
        "keyVersion": "7c73556c521340209371eaf623cc099d"  
      },  
      "keySource": "Microsoft.Keyvault"  
    },  
    ..  
  }  
}
```

Rotation of a customer-managed key

You can rotate a customer-managed key in Azure Key Vault according to your compliance policies. When the key is rotated, you must update the Automation account to use the new key URI.

Rotating the key doesn't trigger re-encryption of secure assets in the Automation account. There's no further action required.

Revocation of access to a customer-managed key

To revoke access to customer-managed keys, use PowerShell or the Azure CLI. For more information, see [Azure Key Vault PowerShell](#) or [Azure Key Vault CLI](#). Revoking access effectively blocks access to all secure assets in the Automation account, as the encryption key is inaccessible by Azure Automation.

Next steps

- To understand Azure Key Vault, see [What is Azure Key Vault?](#).
- To work with certificates, see [Manage certificates in Azure Automation](#).
- To handle credentials, see [Manage credentials in Azure Automation](#).
- To use Automation variables, [Manage variables in Azure Automation](#).
- For help when working with connections, see [Manage connections in Azure Automation](#).

Management of Azure Automation data

7/19/2021 • 5 minutes to read • [Edit Online](#)

This article contains several topics explaining how data is protected and secured in an Azure Automation environment.

TLS 1.2 for Azure Automation

To insure the security of data in transit to Azure Automation, we strongly encourage you to configure the use of Transport Layer Security (TLS) 1.2. The following are a list of methods or clients that communicate over HTTPS to the Automation service:

- Webhook calls
- Hybrid Runbook Workers, which include machines managed by Update Management and Change Tracking and Inventory.
- DSC nodes

Older versions of TLS/Secure Sockets Layer (SSL) have been found to be vulnerable and while they still currently work to allow backwards compatibility, they are **not recommended**. We do not recommend explicitly setting your agent to only use TLS 1.2 unless its necessary, as it can break platform level security features that allow you to automatically detect and take advantage of newer more secure protocols as they become available, such as TLS 1.3.

For information about TLS 1.2 support with the Log Analytics agent for Windows and Linux, which is a dependency for the Hybrid Runbook Worker role, see [Log Analytics agent overview - TLS 1.2](#).

Platform-specific guidance

PLATFORM/LANGUAGE	SUPPORT	MORE INFORMATION
Linux	Linux distributions tend to rely on OpenSSL for TLS 1.2 support.	Check the OpenSSL Changelog to confirm your version of OpenSSL is supported.
Windows 8.0 - 10	Supported, and enabled by default.	To confirm that you are still using the default settings .
Windows Server 2012 - 2016	Supported, and enabled by default.	To confirm that you are still using the default settings
Windows 7 SP1 and Windows Server 2008 R2 SP1	Supported, but not enabled by default.	See the Transport Layer Security (TLS) registry settings page for details on how to enable.

Data retention

When you delete a resource in Azure Automation, it's retained for many days for auditing purposes before permanent removal. You can't see or use the resource during this time. This policy also applies to resources that belong to a deleted Automation account. The retention policy applies to all users and currently can't be customized. However, if you need to keep data for a longer period, you can [forward Azure Automation job data to Azure Monitor logs](#).

The following table summarizes the retention policy for different resources.

DATA	POLICY
Accounts	An account is permanently removed 30 days after a user deletes it.
Assets	An asset is permanently removed 30 days after a user deletes it, or 30 days after a user deletes an account that holds the asset. Assets include variables, schedules, credentials, certificates, Python 2 packages, and connections.
DSC Nodes	A DSC node is permanently removed 30 days after being unregistered from an Automation account using Azure portal or the Unregister-AzAutomationDscNode cmdlet in Windows PowerShell. A node is also permanently removed 30 days after a user deletes the account that holds the node.
Jobs	A job is deleted and permanently removed 30 days after modification, for example, after the job completes, is stopped, or is suspended.
Modules	A module is permanently removed 30 days after a user deletes it, or 30 days after a user deletes the account that holds the module.
Node Configurations/MOF Files	An old node configuration is permanently removed 30 days after a new node configuration is generated.
Node Reports	A node report is permanently removed 90 days after a new report is generated for that node.
Runbooks	A runbook is permanently removed 30 days after a user deletes the resource, or 30 days after a user deletes the account that holds the resource ¹ .

¹The runbook can be recovered within the 30-day window by filing an Azure support incident with Microsoft Azure Support. Go to the [Azure support site](#) and select **Submit a support request**.

Data backup

When you delete an Automation account in Azure, all objects in the account are deleted. The objects include runbooks, modules, configurations, settings, jobs, and assets. They can't be recovered after the account is deleted. You can use the following information to back up the contents of your Automation account before deleting it.

Runbooks

You can export your runbooks to script files using either the Azure portal or the [Get-AzureAutomationRunbookDefinition](#) cmdlet in Windows PowerShell. You can import these script files into another Automation account, as discussed in [Manage runbooks in Azure Automation](#).

Integration modules

You can't export integration modules from Azure Automation, they have to be made available outside of the Automation account.

Assets

You can't export Azure Automation assets: certificates, connections, credentials, schedules, and variables. Instead,

you can use the Azure portal and Azure cmdlets to note the details of these assets. Then use these details to create any assets that are used by runbooks that you import into another Automation account.

You can't retrieve the values for encrypted variables or the password fields of credentials using cmdlets. If you don't know these values, you can retrieve them in a runbook. For retrieving variable values, see [Variable assets in Azure Automation](#). To find out more about retrieving credential values, see [Credential assets in Azure Automation](#).

DSC configurations

You can export your DSC configurations to script files using either the Azure portal or the [Export-AzAutomationDscConfiguration](#) cmdlet in Windows PowerShell. You can import and use these configurations in another Automation account.

Geo-replication in Azure Automation

Geo-replication is standard in Azure Automation accounts. You choose a primary region when setting up your account. The internal Automation geo-replication service assigns a secondary region to the account automatically. The service then continuously backs up account data from the primary region to the secondary region. The full list of primary and secondary regions can be found at [Business continuity and disaster recovery \(BCDR\): Azure Paired Regions](#).

The backup created by the Automation geo-replication service is a complete copy of Automation assets, configurations, and the like. This backup can be used if the primary region goes down and loses data. In the unlikely event that data for a primary region is lost, Microsoft attempts to recover it.

NOTE

Azure Automation stores customer data in the region selected by the customer. For the purpose of BCDR, for all regions except Brazil South and Southeast Asia, Azure Automation data is stored in a different region (Azure paired region). Only for the Brazil South (Sao Paulo State) region of Brazil geography and Southeast Asia region (Singapore) of the Asia Pacific geography, we store Azure Automation data in the same region to accommodate data-residency requirements for these regions.

The Automation geo-replication service isn't accessible directly to external customers if there is a regional failure. If you want to maintain Automation configuration and runbooks during regional failures:

1. Select a secondary region to pair with the geographical region of your primary Automation account.
2. Create an Automation account in the secondary region.
3. In the primary account, export your runbooks as script files.
4. Import the runbooks to your Automation account in the secondary region.

Next steps

- To learn more about secure assets in Azure Automation, see [Encryption of secure assets in Azure Automation](#).
- To find out more about geo-replication, see [Creating and using active geo-replication](#).

Create a standalone Azure Automation account

9/10/2021 • 5 minutes to read • [Edit Online](#)

This article shows you how to create an Azure Automation account using the Azure portal. You can use the Automation account to evaluate and learn about Automation without using additional management features or integrating with Azure Monitor Logs. You can add management features or integrate with Azure Monitor Logs for advanced monitoring of runbook jobs at any point in the future.

With an Automation account, you can authenticate runbooks by managing resources in either Azure Resource Manager or the classic deployment model. One Automation Account can manage resources across all regions and subscriptions for a given tenant.

When you create an Automation account in the Azure portal, the **Run As** account is automatically created. This account does the following tasks:

- Creates a service principal in Azure Active Directory (Azure AD).
- Creates a certificate.
- Assigns the Contributor role, which manages Azure Resource Manager resources by using runbooks.

With this account created for you, you can quickly start building and deploying runbooks to support your automation needs.

Permissions required to create an Automation account

To create or update an Automation account, and to complete the tasks described in this article, you must have the following privileges and permissions:

- To create an Automation account, your Azure AD user account must be added to a role with permissions equivalent to the Owner role for **Microsoft.Automation** resources. For more information, see [Role-Based Access Control in Azure Automation](#).
- In the Azure portal, under **Azure Active Directory > MANAGE > User settings**, if **App registrations** is set to **Yes**, non-administrator users in your Azure AD tenant can [register Active Directory applications](#). If **App registrations** is set to **No**, the user who performs this action must be at a minimum, a member of the Application Developer role in Azure AD.

If you aren't a member of the subscription's Active Directory instance before you're added to the subscription's global Administrator/Co-Administrator role, you're added to Active Directory as a guest. In this scenario, you see this message on the **Add Automation Account** page: **You do not have permissions to create.**

If a user is added to the global Administrator/Co-administrator role first, you can remove the user from the subscription's Active Directory instance. You can readd the user to the User role in Active Directory. To verify user roles:

1. In the Azure portal, go to the Azure Active Directory page.
2. Select **Users and groups**.
3. Select **All users**.
4. After you select a specific user, select **Profile**. The value of the **User type** attribute under the user's profile should not be **Guest**.

Create a new Automation account in the Azure portal

To create an Azure Automation account in the Azure portal, complete the following steps:

1. Sign in to the Azure portal with an account that's a member of the subscription Administrators role and a Co-Administrator of the subscription.
2. Select **+ Create a Resource**.
3. Search for **Automation**. In the search results, select **Automation**.

NAME	PUBLISHER	CATEGORY
Automation	Microsoft	Developer tools
Automation & Control	Microsoft	Monitoring + Manage...
Chef Automate	Chef Software, Inc	Compute

4. On the next screen, select **Create new**.

Add Automation Account

X

* Name i

Enter the account name...

* Subscription

Microsoft Azure



* Resource group

Create new Use existing

* Location

Japan East



* Create Azure Run As account i

Yes

No



The Run As account feature will
create a Run As account and a
Classic Run As account.[Click here](#) to
learn more about Run As accounts.



Learn more about Automation
pricing.



Pin to dashboard

Create

NOTE

If you see the following message in the **Add Automation Account** page, your account is not a member of the subscription Administrators role and a Co-Administrator of the subscription.



You do not have permissions to create a Run As account in Azure Active directory. Please follow the directions in the documentation to learn how to create a Run As account.[Click here to learn more about Run As accounts.](#)

5. On the **Add Automation Account** page, enter a name for your new Automation account in the **Name** field. You can't change this name after it's chosen.

NOTE

Automation account names are unique per region and resource group. Names for deleted Automation accounts might not be immediately available.

6. If you have more than one subscription, use the **Subscription** field to specify the subscription to use for the new account.
7. For **Resource group**, enter or select a new or existing resource group.
8. For **Location**, select an Azure datacenter location.
9. For the **Create Azure Run As account** option, ensure that **Yes** is selected, and then click **Create**.

NOTE

If you choose not to create the Run As account by selecting **No** for **Create Azure Run As account**, a message appears in the **Add Automation Account** page. Although the account is created in the Azure portal, the account doesn't have a corresponding authentication identity in your classic deployment model subscription or in the Azure Resource Manager subscription directory service. Therefore, the Automation account doesn't have access to resources in your subscription. This prevents any runbooks that reference this account from being able to authenticate and perform tasks against resources in those deployment models.



You have chosen not to create a Run As Account. Doing so might block the execution of some runbooks due to lack of access to required resources.[Click here to learn more about Run As accounts.](#)

When the service principal is not created, the Contributor role is not assigned.

10. To track the progress of the Automation account creation, select **Notifications** in the menu.

When the Automation account is successfully created, several resources are automatically created for you. After creation, these runbooks can be safely deleted if you do not wish to keep them. The Run As Accounts, can be used to authenticate to your account in a runbook, and should be left unless you create another one or do not

require them. The following table summarizes resources for the Run As account.

RESOURCE	DESCRIPTION
AzureAutomationTutorial Runbook	An example graphical runbook that demonstrates how to authenticate by using the Run As account. The runbook gets all Resource Manager resources.
AzureAutomationTutorialScript Runbook	An example PowerShell runbook that demonstrates how to authenticate by using the Run As account. The runbook gets all Resource Manager resources.
AzureAutomationTutorialPython2 Runbook	An example Python runbook that demonstrates how to authenticate by using the Run As account. The runbook lists all resource groups present in the subscription.
AzureRunAsCertificate	A certificate asset that's automatically created when the Automation account is created, or by using a PowerShell script for an existing account. The certificate authenticates with Azure so you can manage Azure Resource Manager resources from runbooks. This certificate has a one-year lifespan.
AzureRunAsConnection	A connection asset that's automatically created when the Automation account is created, or by using a PowerShell script for an existing account.

Create a Classic Run As account

Classic Run As accounts are not created by default when you create an Azure Automation account. If you require a Classic Run As account to manage Azure classic resources, perform the following steps:

1. From your Automation account, select **Run As Accounts** under **Account Settings**.
2. Select **Azure Classic Run As Account**.
3. Click **Create** to proceed with Classic Run As account creation.

Next steps

- To learn more about graphical authoring, see [Author graphical runbooks in Azure Automation](#).
- To get started with PowerShell runbooks, see [Tutorial: Create a PowerShell runbook](#).
- To get started with PowerShell Workflow runbooks, see [Tutorial: Create a PowerShell workflow runbook](#).
- To get started with Python 3 runbooks, see [Tutorial: Create a Python 3 runbook](#).
- For a PowerShell cmdlet reference, see [Az.Automation](#).

Create an Azure Automation account using a Resource Manager template

8/27/2021 • 6 minutes to read • [Edit Online](#)

Azure Automation delivers a cloud-based automation and configuration service that supports consistent management across your Azure and non-Azure environments. This article shows you how to deploy an Azure Resource Manager template (ARM template) that creates an Automation account. Using an ARM template takes fewer steps compared to other deployment methods. The JSON template specifies default values for parameters that would likely be used as a standard configuration in your environment. You can store the template in an Azure storage account for shared access in your organization. For more information about working with templates, see [Deploy resources with ARM templates and the Azure CLI](#).

An [ARM template](#) is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

The sample template does the following steps:

- Automates the creation of an Azure Monitor Log Analytics workspace.
- Automates the creation of an Azure Automation account.
- Links the Automation account to the Log Analytics workspace.
- Adds sample Automation runbooks to the account.

NOTE

Creation of the Automation Run As account is not supported when you're using an ARM template. To create a Run As account manually from the portal or with PowerShell, see [Create Run As account](#).

If you don't have an Azure subscription, create a [free account](#) before you begin.

Prerequisites

If you're new to Azure Automation and Azure Monitor, it's important that you understand the configuration details. The understanding can help you avoid errors when you try to create, configure, and use a Log Analytics workspace linked to your new Automation account.

- Review [additional details](#) to fully understand workspace configuration options, such as access control mode, pricing tier, retention, and capacity reservation level.
- Review [workspace mappings](#) to specify the supported regions inline or in a parameter file. Only certain regions are supported for linking a Log Analytics workspace and an Automation account in your subscription.
- If you're new to Azure Monitor Logs and haven't deployed a workspace already, review the [workspace design guidance](#). This document will help you learn about access control, and help you understand the recommended design implementation strategies for your organization.

Review the template

The template used in this article is from [Azure Quickstart Templates](#).

```
{
    "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
        "workspaceName": {
            "type": "string",
            "metadata": {
                "description": "Workspace name"
            }
        },
        "sku": {
            "type": "string",
            "defaultValue": "pergb2018",
            "allowedValues": [
                "pergb2018",
                "Free",
                "Standalone",
                "PerNode",
                "Standard",
                "Premium"
            ],
            "metadata": {
                "description": "Pricing tier: perGB2018 or legacy tiers (Free, Standalone, PerNode, Standard or Premium), which are not available to all customers."
            }
        },
        "dataRetention": {
            "type": "int",
            "defaultValue": 30,
            "minValue": 7,
            "maxValue": 730,
            "metadata": {
                "description": "Number of days to retain data."
            }
        },
        "location": {
            "type": "string",
            "defaultValue": "[resourceGroup().location]",
            "metadata": {
                "description": "Specifies the location in which to create the workspace."
            }
        },
        "automationAccountName": {
            "type": "string",
            "metadata": {
                "description": "Automation account name"
            }
        },
        "sampleGraphicalRunbookName": {
            "type": "String",
            "defaultValue": "AzureAutomationTutorial"
        },
        "sampleGraphicalRunbookDescription": {
            "type": "String",
            "defaultValue": "An example runbook that gets all the Resource Manager resources by using the Run As account (service principal)."
        },
        "samplePowerShellRunbookName": {
            "type": "String",
            "defaultValue": "AzureAutomationTutorialScript"
        },
        "samplePowerShellRunbookDescription": {
            "type": "String",
            "defaultValue": "An example runbook that gets all the Resource Manager resources by using the Run As account (service principal)."
        },
        "samplePython2RunbookName": {
            "type": "String",
            "defaultValue": "AzureAutomationTutorialScript"
        }
    }
}
```

```

        },
        "defaultValue": "AzureAutomationTutorialPython2"
    },
    "samplePython2RunbookDescription": {
        "type": "String",
        "defaultValue": "An example runbook that gets all the Resource Manager resources by using the Run As account (service principal)."
    },
    "_artifactsLocation": {
        "type": "string",
        "defaultValue": "[deployment().properties.templateLink.uri]",
        "metadata": {
            "description": "URI to artifacts location"
        }
    },
    "_artifactsLocationSasToken": {
        "type": "securestring",
        "defaultValue": "",
        "metadata": {
            "description": "The sasToken required to access _artifactsLocation. When the template is deployed using the accompanying scripts, a sasToken will be automatically generated"
        }
    }
},
"resources": [
{
    "type": "Microsoft.OperationalInsights/workspaces",
    "apiVersion": "2020-08-01",
    "name": "[parameters('workspaceName')]",
    "location": "[parameters('location')]",
    "properties": {
        "sku": {
            "name": "[parameters('sku')]"
        },
        "retentionInDays": "[parameters('dataRetention')]",
        "features": {
            "searchVersion": 1,
            "legacy": 0
        }
    }
},
{
    "type": "Microsoft.Automation/automationAccounts",
    "apiVersion": "2020-01-13-preview",
    "name": "[parameters('automationAccountName')]",
    "location": "[parameters('location')]",
    "dependsOn": [
        "[parameters('workspaceName')]"
    ],
    "identity": {
        "type": "SystemAssigned"
    },
    "properties": {
        "sku": {
            "name": "Basic"
        }
    }
},
"resources": [
{
        "type": "runbooks",
        "apiVersion": "2020-01-13-preview",
        "name": "[parameters('sampleGraphicalRunbookName')]",
        "location": "[parameters('location')]",
        "dependsOn": [
            "[parameters('automationAccountName')]"
        ],
        "properties": {
            "runbookType": "GraphPowerShell",
            "logProgress": "false",
            "logVerbose": "false"
        }
    }
]
}
]
```

```

    "logVerbose": "false",
    "description": "[parameters('sampleGraphicalRunbookDescription')]",
    "publishContentLink": {
        "uri": "[uri(parameters('_artifactsLocation'),
concat('scripts/AzureAutomationTutorial.graphrunbook', parameters('_artifactsLocationSasToken')))]",
        "version": "1.0.0.0"
    }
},
{
    "type": "runbooks",
    "apiVersion": "2020-01-13-preview",
    "name": "[parameters('samplePowerShellRunbookName')]",
    "location": "[parameters('location')]",
    "dependsOn": [
        "[parameters('automationAccountName')]"
    ],
    "properties": {
        "runbookType": "PowerShell",
        "logProgress": "false",
        "logVerbose": "false",
        "description": "[parameters('samplePowerShellRunbookDescription')]",
        "publishContentLink": {
            "uri": "[uri(parameters('_artifactsLocation'), concat('scripts/AzureAutomationTutorial.ps1',
parameters('_artifactsLocationSasToken')))]",
            "version": "1.0.0.0"
        }
    }
},
{
    "type": "runbooks",
    "apiVersion": "2020-01-13-preview",
    "name": "[parameters('samplePython2RunbookName')]",
    "location": "[parameters('location')]",
    "dependsOn": [
        "[parameters('automationAccountName')]"
    ],
    "properties": {
        "runbookType": "Python2",
        "logProgress": "false",
        "logVerbose": "false",
        "description": "[parameters('samplePython2RunbookDescription')]",
        "publishContentLink": {
            "uri": "[uri(parameters('_artifactsLocation'),
concat('scripts/AzureAutomationTutorialPython2.py', parameters('_artifactsLocationSasToken')))]",
            "version": "1.0.0.0"
        }
    }
}
],
{
    "type": "Microsoft.OperationalInsights/workspaces/linkedServices",
    "apiVersion": "2020-08-01",
    "name": "[concat(parameters('workspaceName'), '/', 'Automation')]",
    "location": "[parameters('location')]",
    "dependsOn": [
        "[parameters('workspaceName')]",
        "[parameters('automationAccountName')]"
    ],
    "properties": {
        "resourceId": "[resourceId('Microsoft.Automation/automationAccounts',
parameters('automationAccountName'))]"
    }
}
]
}

```

The Azure resources defined in the template:

- **Microsoft.OperationalInsights/workspaces**: creates an Azure Log Analytics workspace.
- **Microsoft.Automation/automationAccounts**: creates an Azure Automation account.
- **Microsoft.Automation/automationAccounts/runbooks**: creates an Azure Automation account runbook.

Deploy the template

1. Select the Deploy to Azure button below to sign in to Azure and open the ARM template.



2. Enter or select the following values:

PROPERTY	DESCRIPTION
Subscription	From the drop-down list, select your Azure subscription.
Resource group	From the drop-down list, select your existing resource group, or select Create new .
Region	This value will autopopulate.
Workspace name	Enter a name for your new Log Analytics Workspace.
Sku	Defaults to the per GB pricing tier released in the April 2018 pricing model. If you want to create or configure a Log Analytics workspace in a subscription that has opted into the April 2018 pricing model, the only valid Log Analytics pricing tier is PerGB2018 .
Data retention	Defaults to 30 days.
Location	The value will autopopulate with the location used for the resource group.
Automation Account name	Enter a name for your new Automation account.
Sample graphical runbook name	Leave as is.
Sample graphical runbook description	Leave as is.
Sample PowerShell runbook name	Leave as is.
Sample PowerShell runbook description	Leave as is.
Sample Python2Runbook name	Leave as is.
Sample Python2Runbook description	Leave as is.
_artifacts Location	Leave as is.* URI to artifacts location.

PROPERTY	DESCRIPTION
_artifacts Location Sas Token	Leave blank. The sasToken required to access <code>_artifactsLocation</code> . When the template is deployed using the accompanying scripts, a <code>sasToken</code> will be automatically generated.

* When you attempt to run the ARM template from PowerShell, CLI, or the Templates feature in the portal, if the `_artifactsLocation` parameter isn't properly set, you'll receive an error message similar to the following:

```
"message": "Deployment template validation failed: 'The template resource '_artifactsLocation' at line '96' and column '31' is not valid: The language expression property 'templateLink' doesn't exist, available properties are 'template, templateHash, parameters, mode, debugSetting, provisioningState'.. Please see https://aka.ms/arm-template-expressions for usage details.'."
```

To prevent this error, when running from the Templates feature in the portal, specify the following value for the `_artifactsLocation` parameter -

```
https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/quickstarts/microsoft.automation/101-automation/azuredetect.json
```

When you run from PowerShell, include the parameter and its value

```
-TemplateUri https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/quickstarts/microsoft.automation/101-automation/azuredetect.json
```

When you run from Azure CLI, include the parameter and its value -

```
--template-uri https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/quickstarts/microsoft.automation/101-automation/azuredetect.json
```

For reference about PowerShell/CLI, see the following - [Create Azure Automation account \(microsoft.com\)](#) under the **Use the template** section.

3. Select **Review + Create** and then **Create**. The deployment can take a few minutes to finish. When completed, the output is similar to the following image:

DeploymentName	:	CreateAutomationAcct	
ResourceGroupName	:	Prod1	
ProvisioningState	:	Succeeded	
Timestamp	:	4/15/2020 3:49:59 PM	
Mode	:	Incremental	
TemplateLink	:		
Parameters	:		
		Name	Type
		workspaceName	String
		sku	String
		dataRetention	Int
		immediatePurgeDataOn30Days	Bool
		location	String
		automationAccountName	String
		automationAccountLocation	String
		sampleGraphicalRunbookName	String
		sampleGraphicalRunbookDescription	String
			Value
		workspaceName	Prod1-LA
		sku	pergb2018
		dataRetention	30
		immediatePurgeDataOn30Days	False
		location	eastus
		automationAccountName	Prod1-AA
		automationAccountLocation	eastus2
		sampleGraphicalRunbookName	AzureAutomationTutorial
		sampleGraphicalRunbookDescription	An example runbook which gets

Review deployed resources

- Once the deployment completes, you'll receive a **Deployment succeeded** notification with a **Go to resource** link. Your **Resource group** page will list your new resources. From the list, select your new Automation account.
- From the left-side, under **Process Automation**, select **Runbooks**. The **Runbooks** page lists the three sample runbooks created with the Automation account.

Name	Authoring status	Runbook type	Last modified
AzureAutomationTutorial	✓ Published	Graphical Runbook	7/23/2020, 10:29 AM
AzureAutomationTutorialPython...	✓ Published	Python 2 Runbook	7/23/2020, 10:29 AM
AzureAutomationTutorialScript	✓ Published	PowerShell Runbook	7/23/2020, 10:29 AM

3. From the left-side, under **Related Resources**, select **Linked workspace**. The **Linked workspace** page shows the Log Analytics workspace you specified earlier that is linked to your Automation account.

This Automation account is linked to the following Log Analytics workspace: [prod1-la](#)

To unlink this Automation account and the Log Analytics workspace you must first remove some solutions. These are the following:

- Update Management
- Change Tracking
- Start/Stop VMs during off-hours

After you remove these solutions you can click **Unlink workspace** above to complete the unlinking process.

If you use the Update Management solution you optionally may want to remove some items

Next steps

[Configure diagnostic settings](#) for your Automation account to send runbook job status and job streams to the linked Log Analytics workspace.

Using a system-assigned managed identity for an Azure Automation account (preview)

9/10/2021 • 8 minutes to read • [Edit Online](#)

This article shows you how to enable a system-assigned managed identity for an Azure Automation account and how to use it to access other resources. For more information on how managed identities work with Azure Automation, see [Managed identities](#).

If you don't have an Azure subscription, create a [free account](#) before you begin.

Prerequisites

- An Azure Automation account. For instructions, see [Create an Azure Automation account](#).
- The latest version of Azure Account modules. Currently this is 2.2.8. (See [Az.Accounts](#) for details about this version.)
- An Azure resource that you want to access from your Automation runbook. This resource needs to have a role defined for the managed identity, which helps the Automation runbook authenticate access to the resource. To add roles, you need to be an owner for the resource in the corresponding Azure AD tenant.
- If you want to execute hybrid jobs using a managed identity, update the Hybrid Runbook Worker to the latest version. The minimum required versions are:
 - Windows Hybrid Runbook Worker: version 7.3.1125.0
 - Linux Hybrid Runbook Worker: version 1.7.4.0

Enable a system-assigned managed identity for an Azure Automation account

Once enabled, the following properties will be assigned to the system-assigned managed identity.

PROPERTY (JSON)	VALUE	DESCRIPTION
principalid	<principal-ID>	The Globally Unique Identifier (GUID) of the service principal object for the system-assigned managed identity that represents your Automation account in the Azure AD tenant. This GUID sometimes appears as an "object ID" or objectID.
tenantid	<Azure-AD-tenant-ID>	The Globally Unique Identifier (GUID) that represents the Azure AD tenant where the Automation account is now a member. Inside the Azure AD tenant, the service principal has the same name as the Automation account.

You can enable a system-assigned managed identity for an Azure Automation account using the Azure portal, PowerShell, the Azure REST API, or ARM template. For the examples involving PowerShell, first sign in to Azure interactively using the [Connect-AzAccount](#) cmdlet and follow the instructions.

```
# Sign in to your Azure subscription
$sub = Get-AzSubscription -ErrorAction SilentlyContinue
if(-not($sub))
{
    Connect-AzAccount -Identity
}

# If you have multiple subscriptions, set the one to use
# Select-AzSubscription -SubscriptionId "<SUBSCRIPTIONID>"
```

Then initialize a set of variables that will be used throughout the examples. Revise the values below and then execute.

```
$subscriptionID = "subscriptionID"
$resourceGroup = "resourceGroupName"
$automationAccount = "automationAccountName"
```

IMPORTANT

The new Automation account-level identity will override any previous VM-level system-assigned identities which are described in [Use runbook authentication with managed identities](#). If you're running hybrid jobs on Azure VMs that use a VM's system-assigned identity to access runbook resources, then the Automation account identity will be used for the hybrid jobs. This means your existing job execution may be affected if you've been using the Customer Managed Keys (CMK) feature of your Automation account.

If you wish to continue using the VM's managed identity, you shouldn't enable the Automation account-level identity. If you've already enabled it, you can disable the Automation account system-assigned managed identity. See [Disable your Azure Automation account managed identity](#).

Enable using the Azure portal

Perform the following steps:

1. Sign in to the [Azure portal](#).
2. In the Azure portal, navigate to your Automation account.
3. Under **Account Settings**, select **Identity**.
4. Set the **System assigned** option to **On** and press **Save**. When you're prompted to confirm, select **Yes**.

Home > automationdemo

The screenshot shows the Azure portal interface for managing an Automation account named 'automationdemo'. On the left, there's a sidebar with various options like Python packages, Credentials, Connections, Certificates, Variables, and several 'Related Resources' such as Linked workspace, Event grid, and Start/Stop VM. The 'Identity' option is highlighted with a grey bar at the bottom of the list. The main content area has a title 'automationdemo | Identity' and a sub-section 'System assigned'. It explains that a system-assigned managed identity is restricted to one per resource (Azure RBAC) and is authenticated with Azure. There are buttons for Save, Discard, Refresh, and Got feedback? Below that is a 'Status' toggle switch, which is currently set to 'On'. At the bottom, there's a section for 'Object ID' containing the value '12345678-1111-2222-3333-1234567891234' and a 'Permissions' section labeled 'Azure role assignments'.

Your Automation account can now use the system-assigned identity, which is registered with Azure Active Directory (Azure AD) and is represented by an object ID.

This screenshot shows the 'Identity' settings page for an Automation account. It features a 'Status' toggle switch set to 'On', an 'Object ID' input field containing the value '12345678-1111-2222-3333-1234567891234', and a 'Permissions' section with a button labeled 'Azure role assignments'.

Enable using PowerShell

Use PowerShell cmdlet [Set-AzAutomationAccount](#) to enable the system-assigned managed identity.

```
$output = Set-AzAutomationAccount ` 
    -ResourceGroupName $resourceGroup ` 
    -Name $automationAccount ` 
    -AssignSystemIdentity

$output
```

The output should look similar to the following:

```

SubscriptionId      : SubscriptionID
ResourceGroupName   : ContosoLab
AutomationAccountName : myAutomationAccount
Location           : westus
State               : Ok
Plan                : Basic
CreationTime       : 7/9/2021 12:18:26 AM +00:00
LastModifiedTime   : 7/9/2021 4:37:06 PM +00:00
LastModifiedBy     :
Tags               : {}
Identity           : Microsoft.Azure.Management.Automation.Models.Identity
Encryption         : Microsoft.Azure.Management.Automation.Models.EncryptionProperties
PublicNetworkAccess :

```

For additional output, execute: `$output.identity | ConvertTo-Json`.

Enable using a REST API

Syntax and example steps are provided below.

Syntax

The body syntax below enables a system-assigned managed identity to an existing Automation account using the **HTTP PATCH** method. However, this syntax will remove any existing user-assigned managed identities associated with the Automation account.

```
{
  "identity": {
    "type": "SystemAssigned"
  }
}
```

If there are multiple user-assigned identities defined, to retain them and only remove the system-assigned identity, you need to specify each user-assigned identity using comma-delimited list. The example below uses the **HTTP PATCH** method.

```
{
  "identity" : {
    "type": "SystemAssigned, UserAssigned",
    "userAssignedIdentities": {
      "/subscriptions/00000000-0000-0000-0000-
      000000000000/resourceGroups/resourceGroupName/providers/Microsoft.ManagedIdentity/userAssignedIdentities/cmk
      ID": {},
      "/subscriptions/00000000-0000-0000-0000-
      000000000000/resourceGroups/resourceGroupName/providers/Microsoft.ManagedIdentity/userAssignedIdentities/cmk
      ID2": {}
    }
  }
}
```

The syntax of the API is as follows:

```
PATCH https://management.azure.com/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/resource-group-
name/providers/Microsoft.Automation/automationAccounts/automation-account-name?api-version=2020-01-13-
preview
```

Example

Perform the following steps.

1. Copy and paste the body syntax into a file named `body_sa.json`. Save the file on your local machine or in an Azure storage account.
2. Update the variable value below and then execute.

```
$file = "path\body_sa.json"
```

3. This example uses the PowerShell cmdlet [Invoke-RestMethod](#) to send the PATCH request to your Automation account.

```
# build URI
$URI =
"https://management.azure.com/subscriptions/$subscriptionID/resourceGroups/$resourceGroup/providers/Microsoft.Automation/automationAccounts/$automationAccount`?api-version=2020-01-13-preview"

# build body
$body = Get-Content $file

# obtain access token
$azContext = Get-AzContext
$azProfile =
[Microsoft.Azure.Commands.Common.Authentication.Abstractions.AzureRmProfileProvider]::Instance.Profile
$profileClient = New-Object -TypeName Microsoft.Azure.Commands.ResourceManager.Common.RMProfileClient
-ArgumentList ($azProfile)
$token = $profileClient.AcquireAccessToken($azContext.Subscription.TenantId)
$authHeader = @{
    'Content-Type'='application/json'
    'Authorization'='Bearer ' + $token.AccessToken
}

# Invoke the REST API
$response = Invoke-RestMethod -Uri $URI -Method PATCH -Headers $authHeader -Body $body

# Review output
$response.identity | ConvertTo-Json
```

The output should look similar to the following:

```
{
    "PrincipalId": "aaaaaaaa-aaaa-aaa-aaa-aaaaaaaaaaaa",
    "TenantId": "bbbbbbbb-bbbb-bbbb-bbbb-bbbbbbbbbb",
    "Type": 0,
    "UserAssignedIdentities": null
}
```

Enable using an ARM template

Syntax and example steps are provided below.

Template syntax

The sample template syntax below enables a system-assigned managed identity to the existing Automation account. However, this syntax will remove any existing user-assigned managed identities associated with the Automation account.

```
{  
    "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",  
    "contentVersion": "1.0.0.0",  
    "resources": [  
        {  
            "type": "Microsoft.Automation/automationAccounts",  
            "apiVersion": "2020-01-13-preview",  
            "name": "yourAutomationAccount",  
            "location": "[resourceGroup().location]",  
            "identity": {  
                "type": "SystemAssigned"  
            },  
            "properties": {  
                "sku": {  
                    "name": "Basic"  
                }  
            }  
        }  
    ]  
}
```

Example

Perform the following steps.

1. Revise the syntax of the template above to use your Automation account and save it to a file named `template_sa.json`.
2. Update the variable value below and then execute.

```
$templateFile = "path\template_sa.json"
```

3. Use PowerShell cmdlet [New-AzResourceGroupDeployment](#) to deploy the template.

```
New-AzResourceGroupDeployment `  
    -Name "SystemAssignedDeployment" `  
    -ResourceGroupName $resourceGroup `  
    -TemplateFile $templateFile
```

The command won't produce an output; however, you can use the code below to verify:

```
(Get-AzAutomationAccount `  
    -ResourceGroupName $resourceGroup `  
    -Name $automationAccount).Identity | ConvertTo-Json
```

The output will look similar to the output shown for the REST API example, above.

Give access to Azure resources by obtaining a token

An Automation account can use its system-assigned managed identity to get tokens to access other resources protected by Azure AD, such as Azure Key Vault. These tokens don't represent any specific user of the application. Instead, they represent the application that's accessing the resource. In this case, for example, the token represents an Automation account.

Before you can use your system-assigned managed identity for authentication, set up access for that identity on the Azure resource where you plan to use the identity. To complete this task, assign the appropriate role to that identity on the target Azure resource.

Follow the principle of least privilege and carefully assign permissions only required to execute your runbook.

For example, if the Automation account is only required to start or stop an Azure VM, then the permissions assigned to the Run As account or managed identity needs to be only for starting or stopping the VM. Similarly, if a runbook is reading from blob storage, then assign read only permissions. This example uses Azure PowerShell to show how to assign the Contributor

This example uses Azure PowerShell to show how to assign the Contributor role in the subscription to the target Azure resource. The Contributor role is used as an example, and may or may not be required in your case.

```
New-AzRoleAssignment ` 
    -ObjectId <automation-Identity-object-id> ` 
    -Scope "/subscriptions/<subscription-id>" ` 
    -RoleDefinitionName "Contributor"
```

Authenticate access with system-assigned managed identity

After you enable the managed identity for your Automation account and give an identity access to the target resource, you can specify that identity in runbooks against resources that support managed identity. For identity support, use the Az cmdlet `Connect-AzAccount` cmdlet. See [Connect-AzAccount](#) in the PowerShell reference.

Replace `SubscriptionID` with your actual subscription ID and then execute the following command:

```
Connect-AzAccount -Identity
$AzureContext = Set-AzContext -SubscriptionId "SubscriptionID"
```

NOTE

If your organization is still using the deprecated AzureRM cmdlets, you can use `Connect-AzureRMAccount -Identity`.

Generate an access token without using Azure cmdlets

For HTTP Endpoints make sure of the following.

- The metadata header must be present and should be set to "true".
- A resource must be passed along with the request, as a query parameter for a GET request and as form data for a POST request.
- The X-IDENTITY-HEADER should be set to the value of the environment variable IDENTITY_HEADER for Hybrid Runbook Workers.
- Content Type for the Post request must be 'application/x-www-form-urlencoded'.

Get Access token for System Assigned Identity using HTTP Get

```
$resource= "?resource=https://management.azure.com/"
$url = $env:IDENTITY_ENDPOINT + $resource
$headers = New-Object "System.Collections.Generic.Dictionary[[String],[String]]"
$headers.Add("X-IDENTITY-HEADER", $env:IDENTITY_HEADER)
$headers.Add("Metadata", "True")
$accessToken = Invoke-RestMethod -Uri $url -Method 'GET' -Headers $headers
Write-Output $accessToken.access_token
```

Get Access token for System Assigned Identity using HTTP Post

```

$url = $env:IDENTITY_ENDPOINT
$headers = New-Object "System.Collections.Generic.Dictionary[[String],[String]]"
$headers.Add("X-IDENTITY-HEADER", $env:IDENTITY_HEADER)
$headers.Add("Metadata", "True")
$body = @{resource='https://management.azure.com/' }
$accessToken = Invoke-RestMethod $url -Method 'POST' -Headers $headers -ContentType 'application/x-www-form-urlencoded' -Body $body
Write-Output $accessToken.access_token

```

Using system-assigned managed identity in Azure PowerShell

For more information, see [Get-AzKeyVaultSecret](#).

```

Write-Output "Connecting to azure via Connect-AzAccount -Identity"
Connect-AzAccount -Identity
Write-Output "Successfully connected with Automation account's Managed Identity"
Write-Output "Trying to fetch value from key vault using MI. Make sure you have given correct access to
Managed Identity"
$secret = Get-AzKeyVaultSecret -VaultName '<KVname>' -Name '<KeyName>'

$ssPtr = [System.Runtime.InteropServices.Marshal]::SecureStringToBSTR($secret.SecretValue)
try {
    $secretValueText = [System.Runtime.InteropServices.Marshal]::PtrToStringBSTR($ssPtr)
    Write-Output $secretValueText
} finally {
    [System.Runtime.InteropServices.Marshal]::ZeroFreeBSTR($ssPtr)
}

```

Using system-assigned managed identity in Python Runbook

```

#!/usr/bin/env python3
import os
import requests
# printing environment variables
endPoint = os.getenv('IDENTITY_ENDPOINT')+"?resource=https://management.azure.com/"
identityHeader = os.getenv('IDENTITY_HEADER')
payload={}
headers = {
    'X-IDENTITY-HEADER': identityHeader,
    'Metadata': 'True'
}
response = requests.request("GET", endPoint, headers=headers, data=payload)
print(response.text)

```

Using system-assigned managed identity to Access SQL Database

For details on provisioning access to an Azure SQL database, see [Provision Azure AD admin \(SQL Database\)](#).

```

$queryParameter = "?resource=https://database.windows.net/"
$url = $env:IDENTITY_ENDPOINT + $queryParameter
$Headers = New-Object "System.Collections.Generic.Dictionary[[String],[String]]"
$Headers.Add("X-IDENTITY-HEADER", $env:IDENTITY_HEADER)
$Headers.Add("Metadata", "True")
$content =[System.Text.Encoding]::Default.GetString((Invoke-WebRequest -UseBasicParsing -Uri $url -Method 'GET' -Headers $Headers).RawContentStream.ToArray()) | ConvertFrom-Json
$Token = $content.access_token
echo "The managed identities for Azure resources access token is $Token"
$SQLServerName = "<ServerName>"      # Azure SQL logical server name
$DatabaseName = "<DBname>"        # Azure SQL database name
Write-Host "Create SQL connection string"
$conn = New-Object System.Data.SqlClient.SqlConnection
$conn.ConnectionString = "Data Source=$SQLServerName.database.windows.net;Initial Catalog=$DatabaseName;Connect Timeout=30"
$conn.AccessToken = $Token
Write-host "Connect to database and execute SQL script"
$conn.Open()
$ddlstmt = "CREATE TABLE Person( PersonId INT IDENTITY PRIMARY KEY, FirstName NVARCHAR(128) NOT NULL)"
Write-host ""
Write-host "SQL DDL command"
$ddlstmt
$command = New-Object -TypeName System.Data.SqlClient.SqlCommand($ddlstmt, $conn)
Write-host "results"
$command.ExecuteNonQuery()
$conn.Close()

```

Next steps

- If your runbooks aren't completing successfully, review [Troubleshoot Azure Automation managed identity issues \(preview\)](#).
- If you need to disable a managed identity, see [Disable your Azure Automation account managed identity \(preview\)](#).
- For an overview of Azure Automation account security, see [Automation account authentication overview](#).

Using a user-assigned managed identity for an Azure Automation account (preview)

9/10/2021 • 7 minutes to read • [Edit Online](#)

This article shows you how to add a user-assigned managed identity for an Azure Automation account and how to use it to access other resources. For more information on how managed identities work with Azure Automation, see [Managed identities](#).

NOTE

User-assigned managed identities are supported for cloud jobs only.

If you don't have an Azure subscription, create a [free account](#) before you begin.

Prerequisites

- An Azure Automation account. For instructions, see [Create an Azure Automation account](#).
- A system-assigned managed identity. For instructions, see [Using a system-assigned managed identity for an Azure Automation account \(preview\)](#).
- A user-assigned managed identity. For instructions, see [Create a user-assigned managed identity](#).
- The user-assigned managed identity and the target Azure resources that your runbook manages using that identity must be in the same Azure subscription.
- The latest version of Azure Account modules. Currently this is 2.2.8. (See [Az.Accounts](#) for details about this version.)
- An Azure resource that you want to access from your Automation runbook. This resource needs to have a role defined for the user-assigned managed identity, which helps the Automation runbook authenticate access to the resource. To add roles, you need to be an owner for the resource in the corresponding Azure AD tenant.
- If you want to execute hybrid jobs using a user-assigned managed identity, update the Hybrid Runbook Worker to the latest version. The minimum required versions are:
 - Windows Hybrid Runbook Worker: version 7.3.1125.0
 - Linux Hybrid Runbook Worker: version 1.7.4.0

Add user-assigned managed identity for Azure Automation account

You can add a user-assigned managed identity for an Azure Automation account using the Azure portal, PowerShell, the Azure REST API, or ARM template. For the examples involving PowerShell, first sign in to Azure interactively using the [Connect-AzAccount](#) cmdlet and follow the instructions.

```

# Sign in to your Azure subscription
$sub = Get-AzSubscription -ErrorAction SilentlyContinue
if(-not($sub))
{
    Connect-AzAccount -Subscription
}

# If you have multiple subscriptions, set the one to use
# Select-AzSubscription -SubscriptionId "<SUBSCRIPTIONID>"
```

Then initialize a set of variables that will be used throughout the examples. Revise the values below and then execute"

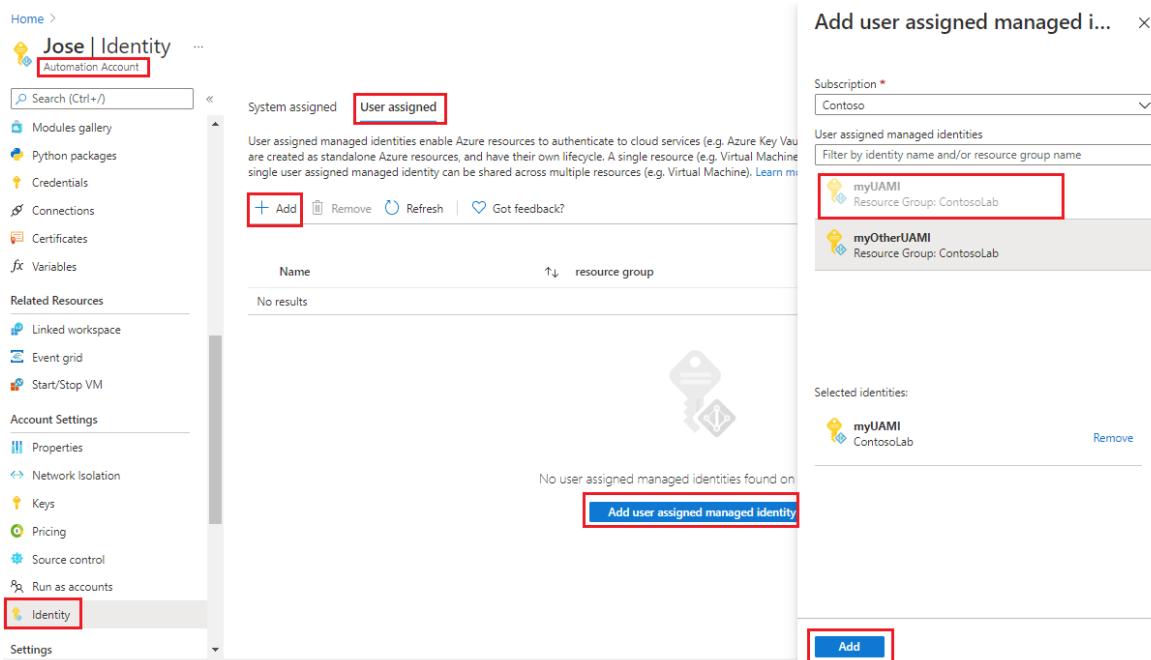
```

$subscriptionID = "subscriptionID"
$resourceGroup = "resourceGroupName"
$automationAccount = "automationAccountName"
$userAssignedOne = "userAssignedIdentityOne"
$userAssignedTwo = "userAssignedIdentityTwo"
```

Add using the Azure portal

Perform the following steps:

1. Sign in to the [Azure portal](#).
2. In the Azure portal, navigate to your Automation account.
3. Under **Account Settings**, select **Identity**.
4. Select the **User assigned** tab, and then select **Add**.
5. Select your existing user-assigned managed identity and then select **Add**. You'll then be returned to the **User assigned** tab.



Add using PowerShell

Use PowerShell cmdlet [Set-AzAutomationAccount](#) to add the user-assigned managed identities. You must first consider whether there's an existing system-assigned managed identity. The example below adds two existing user-assigned managed identities to an existing Automation account, and will disable a system-assigned managed identity if one exists.

```

$output = Set-AzAutomationAccount ` 
    -ResourceGroupName $resourceGroup ` 
    -Name $automationAccount ` 
    -AssignUserIdentity 
"/subscriptions/$subscriptionID/resourcegroups/$resourceGroup/providers/Microsoft.ManagedIdentity/userAssignedIdentities/$userAssignedOne", ` 
"/subscriptions/$subscriptionID/resourcegroups/$resourceGroup/providers/Microsoft.ManagedIdentity/userAssignedIdentities/$userAssignedTwo" 

$output

```

To keep an existing system-assigned managed identity, use the following:

```

$output = Set-AzAutomationAccount ` 
    -ResourceGroupName $resourceGroup ` 
    -Name $automationAccount ` 
    -AssignUserIdentity 
"/subscriptions/$subscriptionID/resourcegroups/$resourceGroup/providers/Microsoft.ManagedIdentity/userAssignedIdentities/$userAssignedOne", ` 
"/subscriptions/$subscriptionID/resourcegroups/$resourceGroup/providers/Microsoft.ManagedIdentity/userAssignedIdentities/$userAssignedTwo" ` 
    -AssignSystemIdentity 

$output

```

The output should look similar to the following:

```

SubscriptionId      : SubscriptionID
ResourceGroupName   : ContosoLab
AutomationAccountName : myAutomationAccount
Location           : westus
State               : Ok
Plan                : Basic
CreationTime        : 7/9/2021 12:18:26 AM +00:00
LastModifiedTime    : 7/9/2021 4:37:06 PM +00:00
LastModifiedBy      : 
Tags                : {}
Identity            : Microsoft.Azure.Management.Automation.Models.Identity
Encryption          : Microsoft.Azure.Management.Automation.Models.EncryptionProperties
PublicNetworkAccess : 

```

For additional output, execute: `$output.identity | ConvertTo-Json`.

Add using a REST API

Syntax and example steps are provided below.

Syntax

The sample body syntax below enables a system-assigned managed identity if not already enabled and assigns two existing user-assigned managed identities to the existing Automation account.

PATCH

```
{
  "identity": {
    "type": "SystemAssigned, UserAssigned",
    "userAssignedIdentities": {
      "/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/resource-group-
name/providers/Microsoft.ManagedIdentity/userAssignedIdentities/firstIdentity": {},
      "/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/resource-group-
name/providers/Microsoft.ManagedIdentity/userAssignedIdentities/secondIdentity": {}
    }
  }
}
```

The syntax of the API is as follows:

```
https://management.azure.com/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/resource-group-name/providers/Microsoft.Automation/automationAccounts/automation-account-name?api-version=2020-01-13-preview
```

Example

Perform the following steps.

1. Revise the syntax of the body above into a file named `body_ua.json`. Save the file on your local machine or in an Azure storage account.
2. Revise the variable value below and then execute.

```
$file = "path\body_ua.json"
```

3. This example uses the PowerShell cmdlet [Invoke-RestMethod](#) to send the PATCH request to your Automation account.

```
# build URI
$URI =
"https://management.azure.com/subscriptions/$subscriptionID/resourceGroups/$resourceGroup/providers/Microsoft.Automation/automationAccounts/$automationAccount`?api-version=2020-01-13-preview"

# build body
$body = Get-Content $file

# obtain access token
$azContext = Get-AzContext
$azProfile =
[Microsoft.Azure.Commands.Common.Authentication.Abstractions.AzureRmProfileProvider]::Instance.Profile
$profileClient = New-Object -TypeName Microsoft.Azure.Commands.ResourceManager.Common.RMProfileClient -ArgumentList ($azProfile)
$token = $profileClient.AcquireAccessToken($azContext.Subscription.TenantId)
$authHeader = @{
  'Content-Type'='application/json'
  'Authorization'='Bearer ' + $token.AccessToken
}

# Invoke the REST API
$response = Invoke-RestMethod -Uri $URI -Method PATCH -Headers $authHeader -Body $body

# Review output
$response.identity | ConvertTo-Json
```

The output should look similar to the following:

```
{  
    "type": "SystemAssigned, UserAssigned",  
    "principalId": "00000000-0000-0000-0000-000000000000",  
    "tenantId": "00000000-0000-0000-0000-000000000000",  
    "userAssignedIdentities": {  
  
        "/subscriptions/ContosoID/resourcegroups/ContosoLab/providers/Microsoft.ManagedIdentity/userAssignedIdentities/ContosoUAMI1": {  
            "PrincipalId": "00000000-0000-0000-0000-000000000000",  
            "ClientId": "00000000-0000-0000-0000-000000000000"  
        },  
  
        "/subscriptions/ContosoID/resourcegroups/ContosoLab/providers/Microsoft.ManagedIdentity/userAssignedIdentities/ContosoUAMI2": {  
            "PrincipalId": "00000000-0000-0000-0000-000000000000",  
            "ClientId": "00000000-0000-0000-0000-000000000000"  
        }  
    }  
}
```

Add using an ARM template

Syntax and example steps are provided below.

Template syntax

The sample template syntax below enables a system-assigned managed identity if not already enabled and assigns two existing user-assigned managed identities to the existing Automation account.

```
{
    "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
        "automationAccountName": {
            "defaultValue": "YourAutomationAccount",
            "type": "String",
            "metadata": {
                "description": "Automation account name"
            }
        },
        "userAssignedOne": {
            "defaultValue": "userAssignedOne",
            "type": "String",
            "metadata": {
                "description": "User-assigned managed identity"
            }
        },
        "userAssignedTwo": {
            "defaultValue": "userAssignedTwo",
            "type": "String",
            "metadata": {
                "description": "User-assigned managed identity"
            }
        }
    },
    "resources": [
        {
            "type": "Microsoft.Automation/automationAccounts",
            "apiVersion": "2020-01-13-preview",
            "name": "[parameters('automationAccountName')]",
            "location": "[resourceGroup().location]",
            "identity": {
                "type": "SystemAssigned, UserAssigned",
                "userAssignedIdentities": {
                    "[resourceID('Microsoft.ManagedIdentity/userAssignedIdentities/', parameters('userAssignedOne'))]":
                    {},
                    "[resourceID('Microsoft.ManagedIdentity/userAssignedIdentities/', parameters('userAssignedTwo'))]":
                    {}
                }
            },
            "properties": {
                "sku": {
                    "name": "Basic"
                },
                "encryption": {
                    "keySource": "Microsoft.Automation",
                    "identity": {}
                }
            }
        }
    ]
}
```

Example

Perform the following steps.

1. Copy and paste the template into a file named `template_ua.json`. Save the file on your local machine or in an Azure storage account.
2. Revise the variable value below and then execute.

```
$templateFile = "path\template_ua.json"
```

3. Use PowerShell cmdlet [New-AzResourceGroupDeployment](#) to deploy the template.

```
New-AzResourceGroupDeployment ` 
-Name "UserAssignedDeployment" ` 
-ResourceGroupName $resourceGroup ` 
-TemplateFile $templateFile ` 
-automationAccountName $automationAccount ` 
-userAssignedOne $userAssignedOne ` 
-userAssignedTwo $userAssignedTwo
```

The command won't produce an output; however, you can use the code below to verify:

```
(Get-AzAutomationAccount ` 
-ResourceGroupName $resourceGroup ` 
-Name $automationAccount).Identity | ConvertTo-Json
```

The output will look similar to the output shown for the REST API example, above.

Give identity access to Azure resources by obtaining a token

An Automation account can use its user-assigned managed identity to obtain tokens to access other resources protected by Azure AD, such as Azure Key Vault. These tokens don't represent any specific user of the application. Instead, they represent the application that is accessing the resource. In this case, for example, the token represents an Automation account.

Before you can use your user-assigned managed identity for authentication, set up access for that identity on the Azure resource where you plan to use the identity. To complete this task, assign the appropriate role to that identity on the target Azure resource.

Follow the principle of least privilege and carefully assign permissions only required to execute your runbook. For example, if the Automation account is only required to start or stop an Azure VM, then the permissions assigned to the Run As account or managed identity needs to be only for starting or stopping the VM. Similarly, if a runbook is reading from blob storage, then assign read only permissions.

This example uses Azure PowerShell to show how to assign the Contributor role in the subscription to the target Azure resource. The Contributor role is used as an example and may or may not be required in your case. Alternatively, you can also assign the role to the target Azure resource in the [Azure portal](#).

```
New-AzRoleAssignment ` 
-ObjectId <automation-Identity-object-id> ` 
-Scope "/subscriptions/<subscription-id>" ` 
-RoleDefinitionName "Contributor"
```

Authenticate access with user-assigned managed identity

After you enable the user-assigned managed identity for your Automation account and give an identity access to the target resource, you can specify that identity in runbooks against resources that support managed identity. For identity support, use the Az cmdlet [Connect-AzAccount](#).

```
Connect-AzAccount-Identity` 
-AccountId<user-assigned-identity-ClientId>
```

Generate an access token without using Azure cmdlets

For HTTP Endpoints make sure of the following.

- The metadata header must be present and should be set to "true".
- A resource must be passed along with the request, as a query parameter for a GET request and as form data for a POST request.
- Content Type for the Post request must be `application/x-www-form-urlencoded`.

Get Access token for user-assigned managed identity using Http Get

```
$resource="?resource=https://management.azure.com/"
$client_id=&client_id=<ClientIdofUSI>
$url=$env:IDENTITY_ENDPOINT+$resource+$client_id
$headers=New-Object"System.Collections.Generic.Dictionary[[String],[String]]"
$headers.Add("Metadata","True")
$accessToken=Invoke-RestMethod-Uri$url-Method'GET' -Headers$headers
Write-Output$accessToken.access_token
```

Get Access token for user-assigned managed identity using Http Post

```
$url=$env:IDENTITY_ENDPOINT
$headers=New-Object"System.Collections.Generic.Dictionary[[String],[String]]"
$headers.Add("Metadata","True")
$body=@{ 'resource'='https://management.azure.com/'
'client_id'='<ClientIdofUSI>' }
$accessToken=Invoke-RestMethod$url-Method'POST' -Headers$headers-ContentType'application/x-www-form-
urlencoded'-Body$body
Write-Output$accessToken.access_token
```

Using user-assigned managed identity in Azure PowerShell

```
Write-Output"ConnectingtoazureviaConnect-AzAccount-Identity -AccountId<ClientIdofUSI>"
Connect-AzAccount-Identity-AccountId<ClientIdofUSI>
Write-Output"SuccessfullyconnectedwithAutomationaccount'sManagedIdentity"
Write-
Output"TryingtorefatchvaluefromkeyvaultusingUserAssignedManagedidentity.MakesureyouhavegivencorrectaccesstoMan
agedIdentity"
$secret=Get-AzKeyVaultSecret-VaultName'<KVname>' -Name'<KeyName>'
$ssPtr=[System.Runtime.InteropServices.Marshal]::SecureStringToBSTR($secret.SecretValue)
try{
$secretValueText=[System.Runtime.InteropServices.Marshal]::PtrToStringBSTR($ssPtr)
Write-Output$secretValueText
}finally{
[System.Runtime.InteropServices.Marshal]::ZeroFreeBSTR($ssPtr)
}
```

Using user-assigned managed identity in Python Runbook

```
#!/usr/bin/envpython3
importos
importrequests

resource="?resource=https://management.azure.com/"
client_id=&client_id=<ClientIdofUSI>
endPoint=os.getenv('IDENTITY_ENDPOINT')+resource+client_id
payload={}
headers={
'Metadata': 'True'
}
response=requests.request("GET",endPoint,headers=headers,data=payload)
print(response.text)
```

Next steps

- If your runbooks aren't completing successfully, review [Troubleshoot Azure Automation managed identity issues \(preview\)](#).
- If you need to disable a managed identity, see [Disable your Azure Automation account managed identity \(preview\)](#).
- For an overview of Azure Automation account security, see [Automation account authentication overview](#).

Disable system-assigned managed identity for Azure Automation account (preview)

7/25/2021 • 2 minutes to read • [Edit Online](#)

You can disable a system-assigned managed identity in Azure Automation by using the Azure portal, or using REST API.

Disable using the Azure portal

You can disable the system-assigned managed identity from the Azure portal no matter how the system-assigned managed identity was originally set up.

1. Sign in to the [Azure portal](#).
2. Navigate to your Automation account and under **Account Settings**, select **Identity**.
3. From the **System assigned** tab, under the **Status** button, select **Off** and then select **Save**. When you're prompted to confirm, select **Yes**.

The system-assigned managed identity is disabled and no longer has access to the target resource.

Disable using REST API

Syntax and example steps are provided below.

Request body

The following request body disables the system-assigned managed identity and removes any user-assigned managed identities using the HTTP PATCH method.

```
{  
  "identity": {  
    "type": "None"  
  }  
}
```

If there are multiple user-assigned identities defined, to retain them and only remove the system-assigned identity, you need to specify each user-assigned identity using comma-delimited list. The example below uses the HTTP PATCH method.

```
{  
  "identity" : {  
    "type": "UserAssigned",  
    "userAssignedIdentities": {  
      "/subscriptions/00000000-0000-0000-0000-  
      00000000/resourceGroups/resourceGroupName/providers/Microsoft.ManagedIdentity/userAssignedIdentities/fir  
      stIdentity": {},  
      "/subscriptions/00000000-0000-0000-0000-  
      00000000/resourceGroups/resourceGroupName/providers/Microsoft.ManagedIdentity/userAssignedIdentities/sec  
      ondIdentity": {}  
    }  
  }  
}
```

The following is the service's REST API request URI to send the PATCH request.

```
PATCH https://management.azure.com/subscriptions/00000000-0000-0000-0000-  
000000000000/resourceGroups/resource-group-  
name/providers/Microsoft.Automation/automationAccounts/automation-account-name?api-version=2020-01-13-  
preview
```

Example

Perform the following steps.

1. Copy and paste the request body, depending on which operation you want to perform, into a file named `body_remove_sa.json`. Save the file on your local machine or in an Azure storage account.
2. Sign in to Azure interactively using the [Connect-AzAccount](#) cmdlet and follow the instructions.

```
# Sign in to your Azure subscription  
$sub = Get-AzSubscription -ErrorAction SilentlyContinue  
if(-not($sub))  
{  
    Connect-AzAccount -Subscription  
}  
  
# If you have multiple subscriptions, set the one to use  
# Select-AzSubscription -SubscriptionId "<SUBSCRIPTIONID>"
```

3. Provide an appropriate value for the variables and then execute the script.

```
$subscriptionID = "subscriptionID"  
$resourceGroup = "resourceGroupName"  
$automationAccount = "automationAccountName"  
$file = "path\body_remove_sa.json"
```

4. This example uses the PowerShell cmdlet [Invoke-RestMethod](#) to send the PATCH request to your Automation account.

```

# build URI
$URI =
"https://management.azure.com/subscriptions/$subscriptionID/resourceGroups/$resourceGroup/providers/M
icrosoft.Automation/automationAccounts/$automationAccount`?api-version=2020-01-13-preview"

# build body
$body = Get-Content $file

# obtain access token
$azContext = Get-AzContext
$azProfile =
[Microsoft.Azure.Commands.Common.Authentication.Abstractions.AzureRmProfileProvider]::Instance.Profil
e
$profileClient = New-Object -TypeName Microsoft.Azure.Commands.ResourceManager.Common.RMProfileClient
-ArgumentList ($azProfile)
$token = $profileClient.AcquireAccessToken($azContext.Subscription.TenantId)
$authHeader = @{
    'Content-Type'='application/json'
    'Authorization'='Bearer ' + $token.AccessToken
}

# Invoke the REST API
Invoke-RestMethod -Uri $URI -Method PATCH -Headers $authHeader -Body $body

# Confirm removal
(Get-AzAutomationAccount `-
-ResourceGroupName $resourceGroup `-
-Name $automationAccount).Identity.Type

```

Depending on the syntax you used, the output will either be: `UserAssigned` or blank.

Next steps

- For more information about enabling managed identities in Azure Automation, see [Enable and use managed identity for Automation \(preview\)](#).
- For an overview of Automation account security, see [Automation account authentication overview](#).

Remove user-assigned managed identity for Azure Automation account (preview)

7/25/2021 • 4 minutes to read • [Edit Online](#)

You can remove a user-assigned managed identity in Azure Automation by using the Azure portal, PowerShell, the Azure REST API, or an Azure Resource Manager (ARM) template.

Remove using the Azure portal

You can remove a user-assigned managed identity from the Azure portal no matter how the user-assigned managed identity was originally added.

1. Sign in to the [Azure portal](#).
2. Navigate to your Automation account and under **Account Settings**, select **Identity**.
3. Select the **User assigned** tab.
4. Select the user-assigned managed identity to be removed from the list.
5. Select **Remove**. When you're prompted to confirm, select **Yes**.

The user-assigned managed identity is removed and no longer has access to the target resource.

Remove using PowerShell

Use PowerShell cmdlet [Set-AzAutomationAccount](#) to remove all user-assigned managed identities and retain an existing system-assigned managed identity.

1. Sign in to Azure interactively using the [Connect-AzAccount](#) cmdlet and follow the instructions.

```
# Sign in to your Azure subscription
$sub = Get-AzSubscription -ErrorAction SilentlyContinue
if(-not($sub))
{
    Connect-AzAccount -Subscription
}
```

2. Provide an appropriate value for the variables and then execute the script.

```
$resourceGroup = "resourceGroupName"
$automationAccount = "automationAccountName"
```

3. Execute [Set-AzAutomationAccount](#).

```
# Removes all UAs, keeps SA
$output = Set-AzAutomationAccount ` 
    -ResourceGroupName $resourceGroup ` 
    -Name $automationAccount ` 
    -AssignSystemIdentity

$output.identity.Type
```

The output will be `SystemAssigned`.

Remove using REST API

You can remove a user-assigned managed identity from the Automation account by using the following REST API call and example.

Request body

Scenario: System-assigned managed identity is enabled or is to be enabled. One of many user-assigned managed identities is to be removed. This example removes a user-assigned managed identity named `firstIdentity` using the HTTP PATCH method.

```
{
  "identity": {
    "type": "SystemAssigned, UserAssigned",
    "userAssignedIdentities": {
      "/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/resource-group-
name/providers/Microsoft.ManagedIdentity/userAssignedIdentities/firstIdentity": null
    }
  }
}
```

Scenario: System-assigned managed identity is enabled or is to be enabled. All user-assigned managed identities are to be removed using the HTTP PUT method.

```
{
  "identity": {
    "type": "SystemAssigned"
  }
}
```

Scenario: System-assigned managed identity is disabled or is to be disabled. One of many user-assigned managed identities is to be removed. This example removes a user-assigned managed identity named `firstIdentity` using the HTTP PATCH method.

```
{
  "identity": {
    "type": "UserAssigned",
    "userAssignedIdentities": {
      "/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/resource-group-
name/providers/Microsoft.ManagedIdentity/userAssignedIdentities/firstIdentity": null
    }
  }
}
```

Scenario: System-assigned managed identity is disabled or is to be disabled. All user-assigned managed identities are to be removed using the HTTP PUT method.

```
{
  "identity": {
    "type": "None"
  }
}
```

The following is the service's REST API request URI to send the PATCH request.

```
https://management.azure.com/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/resource-group-name/providers/Microsoft.Automation/automationAccounts/automation-account-name?api-version=2020-01-13-preview
```

Example

Perform the following steps.

1. Copy and paste the request body, depending on which operation you want to perform, into a file named `body_remove_ua.json`. Make any required modifications, and then save the file on your local machine or in an Azure storage account.
2. Sign in to Azure interactively using the [Connect-AzAccount](#) cmdlet and follow the instructions.

```
# Sign in to your Azure subscription
$sub = Get-AzSubscription -ErrorAction SilentlyContinue
if(-not($sub))
{
    Connect-AzAccount -Subscription
}
```

3. Provide an appropriate value for the variables and then execute the script.

```
$subscriptionID = "subscriptionID"
$resourceGroup = "resourceGroupName"
$automationAccount = "automationAccountName"
$file = "path\body_remove_ua.json"
```

4. This example uses the PowerShell cmdlet [Invoke-RestMethod](#) to send the PATCH request to your Automation account.

```
# build URI
$URI =
"https://management.azure.com/subscriptions/$subscriptionID/resourceGroups/$resourceGroup/providers/Microsoft.Automation/automationAccounts/$automationAccount`?api-version=2020-01-13-preview"

# build body
$body = Get-Content $file

# obtain access token
$azContext = Get-AzContext
$azProfile =
[Microsoft.Azure.Commands.Common.Authentication.Abstractions.AzureRmProfileProvider]::Instance.Profile
$profileClient = New-Object -TypeName Microsoft.Azure.Commands.ResourceManager.Common.RMProfileClient -ArgumentList ($azProfile)
$token = $profileClient.AcquireAccessToken($azContext.Subscription.TenantId)
$authHeader = @{
    'Content-Type'='application/json'
    'Authorization'='Bearer ' + $token.AccessToken
}

# Invoke the REST API
Invoke-RestMethod -Uri $URI -Method PATCH -Headers $authHeader -Body $body

# Confirm removal
(Get-AzAutomationAccount `-
-ResourceGroupName $resourceGroup `-
-Name $automationAccount).Identity.Type
```

Depending on the syntax you used, the output will either be: `SystemAssignedUserAssigned`, `SystemAssigned`, `UserAssigned`, or blank.

Remove using Azure Resource Manager template

If you added the user-assigned managed identity for your Automation account using an Azure Resource Manager template, you can remove the user-assigned managed identity by modifying the template, and then re-running it.

Scenario: System-assigned managed identity is enabled or is to be enabled. One of two user-assigned managed identities is to be removed. This syntax snippet removes **all** user-assigned managed identities **except for** the one passed as a parameter to the template.

```
...
"identity": {
    "type": "SystemAssigned, UserAssigned",
    "userAssignedIdentities": [
        "[resourceID('Microsoft.ManagedIdentity/userAssignedIdentities/',parameters('userAssignedOne'))]"
    ]
},
...
}
```

Scenario: System-assigned managed identity is enabled or is to be enabled. All user-assigned managed identities are to be removed.

```
...
"identity": {
    "type": "SystemAssigned"
},
...
}
```

Scenario: System-assigned managed identity is disabled or is to be disabled. One of two user-assigned managed identities is to be removed. This syntax snippet removes **all** user-assigned managed identities **except for** the one passed as a parameter to the template.

```
...
"identity": {
    "type": "UserAssigned",
    "userAssignedIdentities": [
        "[resourceID('Microsoft.ManagedIdentity/userAssignedIdentities/',parameters('userAssignedOne'))]"
    ]
},
...
}
```

Use the [Get-AzAutomationAccount](#) cmdlet to verify. Depending on the syntax you used, the output will either be: `SystemAssignedUserAssigned`, `SystemAssigned`, or `UserAssigned`.

```
(Get-AzAutomationAccount ` 
    -ResourceGroupName $resourceGroup ` 
    -Name $automationAccount).Identity.Type
```

Next steps

- For more information about enabling managed identities in Azure Automation, see [Enable and use](#)

managed identity for Automation (preview).

- For an overview of Automation account security, see [Automation account authentication overview](#).

Troubleshoot Azure Automation managed identity issues (preview)

7/13/2021 • 2 minutes to read • [Edit Online](#)

This article discusses solutions to problems that you might encounter when you use a managed identity with your Automation account. For general information about using managed identity with Automation accounts, see [Azure Automation account authentication overview](#).

Scenario: Fail to get MSI token for account

Issue

When working with a user-assigned managed identity in your Automation account, you receive an error similar to: Failed to get MSI token for account a123456b-1234-12a3-123a-aa123456aa0b.

Cause

Using a user-assigned managed identity before enabling a system-assigned managed identity for your Automation account.

Resolution

Enable a system-assigned managed identity for your Automation account. Then use the user-assigned managed identity.

Scenario: Attempt to use managed identity with Automation account fails

Issue

When you try to work with managed identities in your Automation account, you encounter an error like this:

```
Connect-AzureRMAccount : An error occurred while sending the request. At line:2 char:1 + Connect-
AzureRMAccount -Identity +
CategoryInfo : CloseError: (:) [Connect-AzureRmAccount], HttpRequestException + FullyQualifiedErrorId :
Microsoft.Azure.Commands.Profile.ConnectAzureRmAccountCommand
```

Cause

The most common cause for this is that you didn't enable the identity before trying to use it. To verify this, run the following PowerShell runbook in the affected Automation account.

```

resource= "?resource=https://management.azure.com/"
$url = $env:IDENTITY_ENDPOINT + $resource
$Headers = New-Object "System.Collections.Generic.Dictionary[[String],[String]]"
$Headers.Add("X-IDENTITY-HEADER", $env:IDENTITY_HEADER)
$Headers.Add("Metadata", "True")

try
{
    $Response = Invoke-RestMethod -Uri $url -Method 'GET' -Headers $Headers
}
catch
{
    $StatusCode = $_.Exception.Response.StatusCode.value_
    $stream = $_.Exception.Response.GetResponseStream()
    $reader = New-Object System.IO.StreamReader($stream)
    $responseBody = $reader.ReadToEnd()

    Write-Output "Request Failed with Status: $StatusCode, Message: $responseBody"
}

```

If the issue is that you didn't enable the identity before trying to use it, you should see a result similar to this:

```
Request Failed with Status: 400, Message: {"Message":"No managed identity was found for Automation account xxxxxxxxxxxx"}
```

Resolution

You must enable an identity for your Automation account before you can use the managed identity service. See [Enable a managed identity for your Azure Automation account \(preview\)](#)

Next steps

If this article doesn't resolve your issue, try one of the following channels for additional support:

- Get answers from Azure experts through [Azure Forums](#).
- Connect with [@AzureSupport](#). This is the official Microsoft Azure account for connecting the Azure community to the right resources: answers, support, and experts.
- File an Azure support incident. Go to the [Azure support site](#), and select **Get Support**.

How to create an Azure Automation Run As account

9/10/2021 • 4 minutes to read • [Edit Online](#)

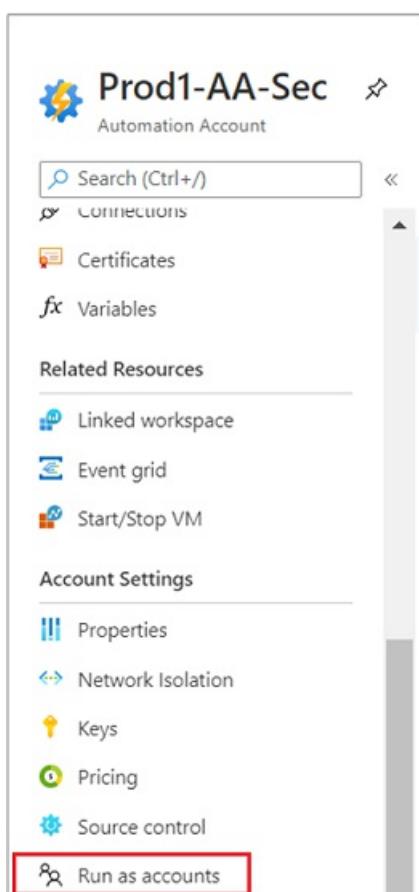
Run As accounts in Azure Automation provide authentication for managing resources on the Azure Resource Manager or Azure Classic deployment model using Automation runbooks and other Automation features. This article describes how to create a Run As or Classic Run As account from the Azure portal or Azure PowerShell.

When you create the Run As or Classic Run As account in the Azure portal, by default it uses a self-signed certificate. If you want to use a certificate issued by your enterprise or third-party certification authority (CA), can use the [PowerShell script to create a Run As account](#).

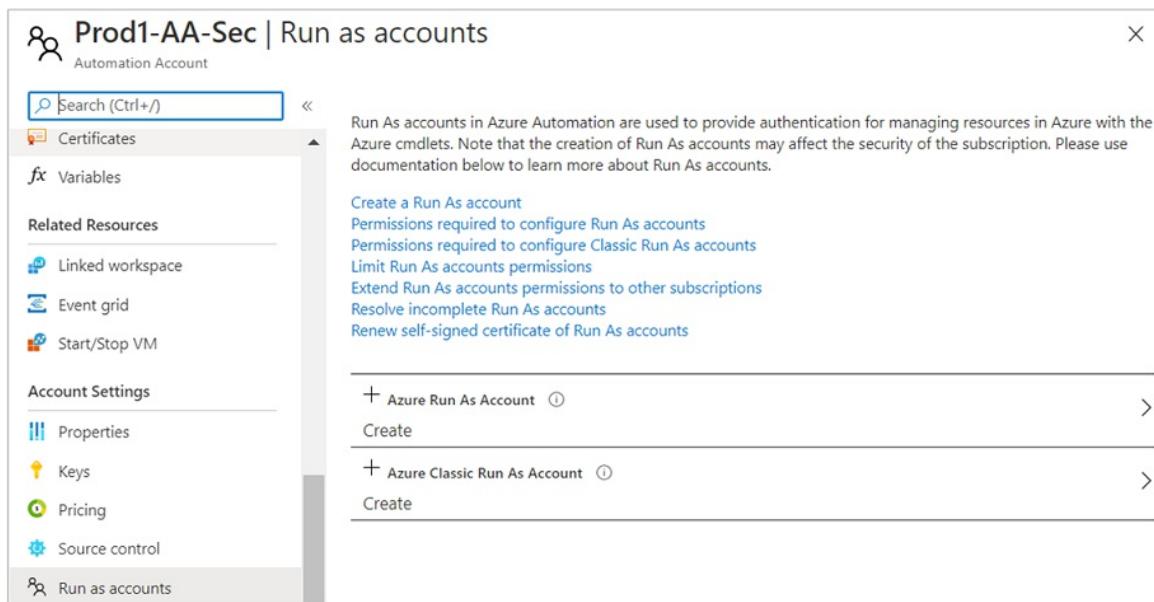
Create account in Azure portal

Perform the following steps to update your Azure Automation account in the Azure portal. The Run As and Classic Run As accounts are created separately. If you don't need to manage classic resources, you can just create the Azure Run As account.

1. Sign in to the Azure portal with an account that is a member of the Subscription Admins role and co-administrator of the subscription.
2. Search for and select **Automation Accounts**.
3. On the **Automation Accounts** page, select your Automation account from the list.
4. In the left pane, select **Run As Accounts** in the **Account Settings** section.



- Depending on the account you require, use the **+ Azure Run As Account** or **+ Azure Classic Run As Account** pane. After reviewing the overview information, click **Create**.



- While Azure creates the Run As account, you can track the progress under **Notifications** from the menu. A banner is also displayed stating that the account is being created. The process can take a few minutes to complete.

Create account using PowerShell

The following list provides the requirements to create a Run As account in PowerShell using a provided script. These requirements apply to both types of Run As accounts.

- Windows 10 or Windows Server 2016 with Azure Resource Manager modules 3.4.1 and later. The PowerShell script doesn't support earlier versions of Windows.
- Azure PowerShell PowerShell 6.2.4 or later. For information, see [How to install and configure Azure PowerShell](#).
- An Automation account, which is referenced as the value for the `AutomationAccountName` and `ApplicationDisplayName` parameters.
- Permissions equivalent to the ones listed in [Required permissions to configure Run As accounts](#).

If you are planning to use a certificate from your enterprise or third-party certificate authority (CA), Automation requires the certificate to have the following configuration:

- Specify the provider **Microsoft Enhanced RSA and AES Cryptographic Provider**
- Marked as exportable
- Configured to use the SHA256 algorithm
- Saved in the `*.pfx` or `*.cer` format.

To get the values for `AutomationAccountName`, `SubscriptionId`, and `ResourceGroupName`, which are required parameters for the PowerShell script, complete the following steps.

- Sign in to the Azure portal.
- Search for and select **Automation Accounts**.
- On the Automation Accounts page, select your Automation account from the list.
- In the left pane, select **Properties**.
- Note the values for **Name**, **Subscription ID**, and **Resource Group** on the **Properties** page.

Name
Prod1-AA-Sec

Subscription
Contoso

Subscription ID
dc54e1c8-8b0b-47ff-856b-718d2cc6f968

Created
9/28/2020, 3:07 PM

Last modified
12/24/2020, 8:32 AM

Resource group
Prod1

Region
eastus2

PowerShell script to create a Run As account

The PowerShell script includes support for several configurations.

- Create a Run As account and/or a Classic Run As account by using a self-signed certificate.
- Create a Run As account and/or a Classic Run As account by using a certificate issued by your enterprise or third-party certification authority (CA).
- Create a Run As account and/or a Classic Run As account by using a self-signed certificate in the Azure Government cloud.

1. Download and save the script to a local folder using the following command.

```
 wget https://raw.githubusercontent.com/azureautomation/runbooks/master/Utility/AzRunAs/Create-RunAsAccount.ps1 -outfile Create-RunAsAccount.ps1
```

2. Start PowerShell with elevated user rights and navigate to the folder that contains the script.

3. Run one of the following commands to create a Run As and/or Classic Run As account based on your requirements.

- Create a Run As account using a self-signed certificate.

```
.\Create-RunAsAccount.ps1 -ResourceGroup <ResourceGroupName> -AutomationAccountName
<NameofAutomationAccount> -SubscriptionId <SubscriptionId> -ApplicationDisplayName
<DisplayNameofAADApplication> -SelfSignedCertPlainPassword <StrongPassword> -
CreateClassicRunAsAccount $false
```

- Create a Run As account and a Classic Run As account by using a self-signed certificate.

```
.\Create-RunAsAccount.ps1 -ResourceGroup <ResourceGroupName> -AutomationAccountName
<NameofAutomationAccount> -SubscriptionId <SubscriptionId> -ApplicationDisplayName
<DisplayNameofAADApplication> -SelfSignedCertPlainPassword <StrongPassword> -
CreateClassicRunAsAccount $true
```

- Create a Run As account and a Classic Run As account by using an enterprise certificate.

```
.\\Create-RunAsAccount.ps1 -ResourceGroup <ResourceGroupName> -AutomationAccountName  
<NameofAutomationAccount> -SubscriptionId <SubscriptionId> -ApplicationDisplayName  
<DisplaynameofAADApplication> -SelfSignedCertPlainPassword <StrongPassword> -  
CreateClassicRunAsAccount $true -EnterpriseCertPathForRunAsAccount  
<EnterpriseCertPfxPathForRunAsAccount> -EnterpriseCertPlainPasswordForRunAsAccount  
<StrongPassword> -EnterpriseCertPathForClassicRunAsAccount  
<EnterpriseCertPfxPathForClassicRunAsAccount> -  
EnterpriseCertPlainPasswordForClassicRunAsAccount <StrongPassword>
```

If you've created a Classic Run As account with an enterprise public certificate (.cer file), use this certificate. The script creates and saves it to the temporary files folder on your computer, under the user profile `%USERPROFILE%\AppData\Local\Temp` you used to execute the PowerShell session. See [Uploading a management API certificate to the Azure portal](#).

- Create a Run As account and a Classic Run As account by using a self-signed certificate in the Azure Government cloud

```
.\\Create-RunAsAccount.ps1 -ResourceGroup <ResourceGroupName> -AutomationAccountName  
<NameofAutomationAccount> -SubscriptionId <SubscriptionId> -ApplicationDisplayName  
<DisplaynameofAADApplication> -SelfSignedCertPlainPassword <StrongPassword> -  
CreateClassicRunAsAccount $true -EnvironmentName AzureUSGovernment
```

4. After the script has executed, you're prompted to authenticate with Azure. Sign in with an account that's a member of the subscription administrators role. If you are creating a Classic Run As account, your account must be a co-administrator of the subscription.

Next steps

- To get started with PowerShell runbooks, see [Tutorial: Create a PowerShell runbook](#).
- To get started with a Python 3 runbook, see [Tutorial: Create a Python 3 runbook](#).

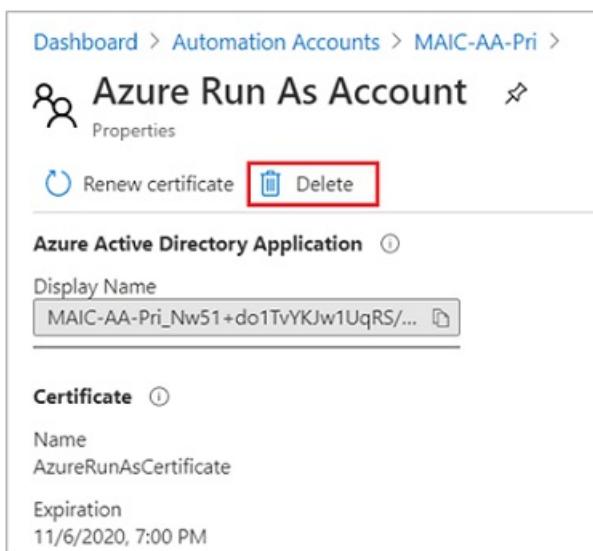
Delete an Azure Automation Run As account

3/5/2021 • 2 minutes to read • [Edit Online](#)

Run As accounts in Azure Automation provide authentication for managing resources on the Azure Resource Manager or Azure Classic deployment model using Automation runbooks and other Automation features. This article describes how to delete a Run As or Classic Run As account. When you perform this action, the Automation account is retained. After you delete the Run As account, you can re-create it in the Azure portal or with the provided PowerShell script.

Delete a Run As or Classic Run As account

1. In the Azure portal, open the Automation account.
2. In the left pane, select **Run As Accounts** in the account settings section.
3. On the Run As Accounts properties page, select either the Run As account or Classic Run As account that you want to delete.
4. On the Properties pane for the selected account, click **Delete**.



The screenshot shows the 'Azure Run As Account' properties pane. At the top, there's a 'Properties' link and a 'Delete' button which is highlighted with a red box. Below that, there's a section for 'Azure Active Directory Application' with a 'Display Name' field containing 'MAIC-AA-Pri_Nw51+do1TvYKJw1UqRS/...'. Under 'Certificate', there's a 'Name' field with 'AzureRunAsCertificate' and an 'Expiration' date of '11/6/2020, 7:00 PM'.

5. While the account is being deleted, you can track the progress under **Notifications** from the menu.

Next steps

To recreate your Run As or Classic Run As account, see [Create Run As accounts](#).

Manage an Azure Automation Run As account

8/2/2021 • 6 minutes to read • [Edit Online](#)

Run As accounts in Azure Automation provide authentication for managing resources on the Azure Resource Manager or Azure Classic deployment model using Automation runbooks and other Automation features.

In this article we cover how to manage a Run as or Classic Run As account, including:

- How to renew a self-signed certificate
- How to renew a certificate from an enterprise or third-party certificate authority (CA)
- Manage permissions for the Run As account

To learn more about Azure Automation account authentication, permissions required to manage the Run as account, and guidance related to process automation scenarios, see [Automation Account authentication overview](#).

Renew a self-signed certificate

The self-signed certificate that you have created for the Run As account expires one year from the date of creation. At some point before your Run As account expires, you must renew the certificate. You can renew it any time before it expires.

When you renew the self-signed certificate, the current valid certificate is retained to ensure that any runbooks that are queued up or actively running, and that authenticate with the Run As account, aren't negatively affected. The certificate remains valid until its expiration date.

NOTE

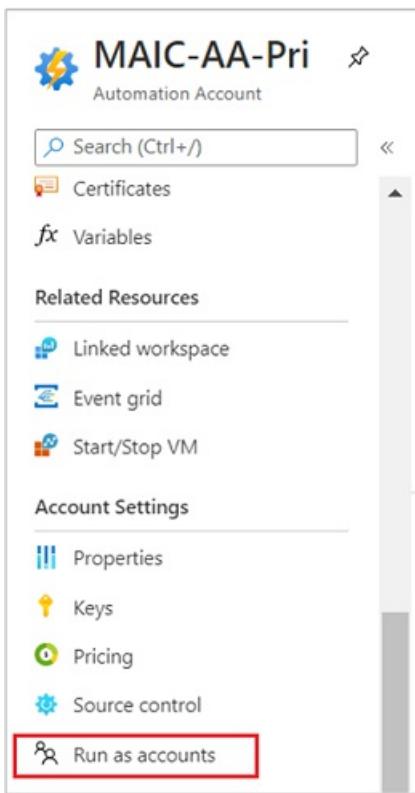
If you think that the Run As account has been compromised, you can delete and re-create the self-signed certificate.

NOTE

If you have configured your Run As account to use a certificate issued by your enterprise or third-party CA and you use the option to renew a self-signed certificate option, the enterprise certificate is replaced by a self-signed certificate. To renew your certificate in this case, see [Renew an enterprise or third-party certificate](#).

Use the following steps to renew the self-signed certificate.

1. Sign-in to the [Azure portal](#).
2. Go to your Automation account and select **Run As Accounts** in the account settings section.



3. On the **Run As Accounts** properties page, select either **Run As Account** or **Classic Run As Account** depending on which account you need to renew the certificate for.
4. On the **Properties** page for the selected account, select **Renew certificate**.

This screenshot shows the 'Azure Run As Account' properties page. At the top, there are two buttons: 'Renew certificate' (highlighted with a red box) and 'Delete'. Below these, under 'Azure Active Directory Application', is a 'Display Name' field containing 'MAIC-AA-Pri_Nw51+do1TvYKJw1UqRS...'. Under 'Certificate', it shows a 'Name' of 'AzureRunAsCertificate' and an 'Expiration' date of '11/6/2020, 7:00 PM'.

5. While the certificate is being renewed, you can track the progress under **Notifications** from the menu.

Renew an enterprise or third-party certificate

Every certificate has a built-in expiration date. If the certificate you assigned to the Run As account was issued by a certification authority (CA), you need to perform a different set of steps to configure the Run As account with the new certificate before it expires. You can renew it any time before it expires.

1. Import the renewed certificate following the steps for [Create a new certificate](#). Automation requires the certificate to have the following configuration:
 - Specify the provider **Microsoft Enhanced RSA and AES Cryptographic Provider**
 - Marked as exportable
 - Configured to use the SHA256 algorithm

- Saved in the `*.pfx` or `*.cer` format.

After you import the certificate, note or copy the certificate **Thumbprint** value. This value is used to update the Run As connection properties with the new certificate.

2. Sign in to the [Azure portal](#).
3. Search for and select **Automation Accounts**.
4. On the Automation Accounts page, select your Automation account from the list.
5. In the left pane, select **Connections**.
6. On the **Connections** page, select **AzureRunAsConnection** and update the **Certificate Thumbprint** with the new certificate thumbprint.
7. Select **Save** to commit your changes.

Grant Run As account permissions in other subscriptions

Azure Automation supports using a single Automation account from one subscription, and executing runbooks against Azure Resource Manager resources across multiple subscriptions. This configuration does not support the Azure Classic deployment model.

You assign the Run As account service principal the [Contributor](#) role in the other subscription, or more restrictive permissions. For more information, see [Role-based access control](#) in Azure Automation. To assign the Run As account to the role in the other subscription, the user account performing this task needs to be a member of the [Owner](#) role in that subscription.

NOTE

This configuration only supports multiple subscriptions of an organization using a common Azure AD tenant.

Before granting the Run As account permissions, you need to first note the display name of the service principal to assign.

1. Sign in to the [Azure portal](#).
2. From your Automation account, select **Run As Accounts** under **Account Settings**.
3. Select **Azure Run As Account**.
4. Copy or note the value for **Display Name** on the **Azure Run As Account** page.

For detailed steps for how to add role assignments, check out the following articles depending on the method you want to use.

- [Assign Azure roles using the Azure portal](#)
- [Assign Azure roles using Azure PowerShell](#)
- [Assign Azure roles using the Azure CLI](#)
- [Assign Azure roles using the REST API](#)

After assigning the Run As account to the role, in your runbook specify

```
Set-AzContext -SubscriptionId "xxxx-xxxx-xxxx-xxxx"
```

to set the subscription context to use. For more information, see [Set-AzContext](#).

Limit Run As account permissions

To control the targeting of Automation against resources in Azure, you can run the [Update-AutomationRunAsAccountRoleAssignments.ps1](#) script. This script changes your existing Run As account service

principal to create and use a custom role definition. The role has permissions for all resources except [Key Vault](#).

IMPORTANT

After you run the `Update-AutomationRunAsAccountRoleAssignments.ps1` script, runbooks that access Key Vault through the use of Run As accounts no longer work. Before running the script, you should review runbooks in your account for calls to Azure Key Vault. To enable access to Key Vault from Azure Automation runbooks, you must [add the Run As account to Key Vault's permissions](#).

If you need to further restrict what the Run As service principal can do, you can add other resource types to the `NotActions` element of the custom role definition. The following example restricts access to `Microsoft.Compute/*`. If you add this resource type to `NotActions` for the role definition, the role will not be able to access any Compute resource. To learn more about role definitions, see [Understand role definitions for Azure resources](#).

```
$roleDefinition = Get-AzRoleDefinition -Name 'Automation RunAs Contributor'  
$roleDefinition.NotActions.Add("Microsoft.Compute/*")  
$roleDefinition | Set-AzRoleDefinition
```

You can determine if the service principal used by your Run As account assigned the **Contributor** role or a custom one.

1. Sign in to the [Azure portal](#).
2. Go to your Automation account and select **Run As Accounts** in the account settings section.
3. Select **Azure Run As Account**.
4. Select **Role** to locate the role definition that is being used.

Role	Assigned at
Contributor	Microsoft Azure

You can also determine the role definition used by the Run As accounts for multiple subscriptions or Automation accounts. Do this by using the `Check-AutomationRunAsAccountRoleAssignments.ps1` script in the PowerShell Gallery.

Add permissions to Key Vault

You can allow Azure Automation to verify if Key Vault and your Run As account service principal are using a custom role definition. You must:

- Grant permissions to Key Vault.
- Set the access policy.

You can use the `Extend-AutomationRunAsAccountRoleAssignmentToKeyVault.ps1` script in the PowerShell Gallery to grant your Run As account permissions to Key Vault. See [Assign a Key Vault access policy](#) for more details on setting permissions on Key Vault.

Resolve misconfiguration issues for Run As accounts

Some configuration items necessary for a Run As or Classic Run As account might have been deleted or created

improperly during initial setup. Possible instances of misconfiguration include:

- Certificate asset
- Connection asset
- Run As account removed from the Contributor role
- Service principal or application in Azure AD

For such misconfiguration instances, the Automation account detects the changes and displays a status of *Incomplete* on the Run As Accounts properties pane for the account.

The screenshot shows the 'Run as accounts' section of the Azure Automation portal. On the left, there's a sidebar with 'Related Resources' (Linked workspace, Event grid, Start/Stop VM) and 'Account Settings' (Properties, Keys, Pricing, Source control, Run as accounts). The 'Run as accounts' item is highlighted. The main area has a search bar and a note about Run As accounts. Below the note is a list of actions: Create a Run As account, Permissions required to configure Run As accounts, Permissions required to configure Classic Run As accounts, Limit Run As accounts permissions, Extend Run As accounts permissions to other subscriptions, Resolve incomplete Run As accounts, and Renew self-signed certificate of Run As accounts. Under this list, two accounts are listed: 'Azure Run As Account' with status 'Incomplete' and 'Azure Classic Run As Account' with status 'Incomplete' and expiration date 'Expires 6/18/2021, 8:00 PM'.

When you select the Run As account, the account properties pane displays the following error message:

The Run As account is incomplete. Either one of these was deleted or not created - Azure Active Directory Application, Service Principal, Role, Automation Certificate asset, Automation Connect asset - or the Thumbprint is not identical between Certificate and Connection. Please delete and then re-create the Run As Account.

You can quickly resolve these Run As account issues by [deleting](#) and [re-creating](#) the Run As account.

Next steps

- [Application Objects and Service Principal Objects](#).
- [Certificates overview for Azure Cloud Services](#).
- To create or re-create a Run As account, see [Create a Run As account](#).
- If you no longer need to use a Run As account, see [Delete a Run As account](#).

Authenticate runbooks with Amazon Web Services

11/2/2020 • 2 minutes to read • [Edit Online](#)

Automating common tasks with resources in Amazon Web Services (AWS) can be accomplished with Automation runbooks in Azure. You can automate many tasks in AWS using Automation runbooks just like you can with resources in Azure. For authentication, you must have an Azure subscription.

Obtain AWS subscription and credentials

To authenticate with AWS, you must obtain an AWS subscription and specify a set of AWS credentials to authenticate your runbooks running from Azure Automation. Specific credentials required are the AWS Access Key and Secret Key. See [Using AWS Credentials](#).

Configure Automation account

You can use an existing Automation account to authenticate with AWS. Alternatively, you can dedicate an account for runbooks targeting AWS resources. In this case, create a new [Automation account](#).

Store AWS credentials

You must store the AWS credentials as assets in Azure Automation. See [Managing Access Keys for your AWS Account](#) for instructions on creating the Access Key and the Secret Key. When the keys are available, copy the Access Key ID and the Secret Key ID in a safe place. You can download your key file to store it somewhere safe.

Create credential asset

After you have created and copied your AWS security keys, you must create a Credential asset with the Automation account. The asset allows you to securely store the AWS keys and reference them in your runbooks. See [Create a new credential asset with the Azure portal](#). Enter the following AWS information in the fields provided:

- **Name** - **AWScred**, or an appropriate value following your naming standards
- **User name** - Your access ID
- **Password** - Name of your Secret Key

Next steps

- To learn how to create runbooks to automate tasks in AWS, see [Deploy an Amazon Web Services VM with a runbook](#).

Use Azure AD to authenticate to Azure

6/21/2021 • 4 minutes to read • [Edit Online](#)

The [Azure Active Directory \(AD\)](#) service enables a number of administrative tasks, such as user management, domain management, and single sign-on configuration. This article describes how to use Azure AD within Azure Automation as the provider for authentication to Azure.

Install Azure AD modules

You can enable Azure AD through the following PowerShell modules:

- Azure Active Directory PowerShell for Graph (AzureRM and Az modules). Azure Automation ships with the AzureRM module and its recent upgrade, the Az module. Functionality includes non-interactive authentication to Azure using Azure AD user (OrgId) credential-based authentication. See [Azure AD 2.0.2.76](#).
- Microsoft Azure Active Directory for Windows PowerShell (MSOnline module). This module enables interactions with Microsoft Online, including Microsoft 365.

NOTE

PowerShell Core does not support the MSOnline module. To use the module cmdlets, you must run them from Windows PowerShell. You're encouraged to use the newer Azure Active Directory PowerShell for Graph modules instead of the MSOnline module.

Preinstallation

Before installing the Azure AD modules on your computer:

- Uninstall any previous versions of the AzureRM/Az module and the MSOnline module.
- Uninstall the Microsoft Online Services Sign-In Assistant to ensure correct operation of the new PowerShell modules.

Install the AzureRM and Az modules

NOTE

To work with these modules, you must use PowerShell version 5.1 or later with a 64-bit version of Windows.

1. Install Windows Management Framework (WMF) 5.1. See [Install and Configure WMF 5.1](#).
2. Install AzureRM and/or Az using instructions in [Install Azure PowerShell on Windows with PowerShellGet](#).

Install the MSOnline module

NOTE

To install the MSOnline module, you must be a member of an admin role. See [About admin roles](#).

1. Ensure that the Microsoft .NET Framework 3.5.x feature is enabled on your computer. It's likely that your computer has a newer version installed, but backward compatibility with older versions of the .NET Framework can be enabled or disabled.

2. Install the 64-bit version of the [Microsoft Online Services Sign-in Assistant](#).
3. Run Windows PowerShell as an administrator to create an elevated Windows PowerShell command prompt.
4. Deploy Azure Active Directory from [MSOnline 1.0](#).
5. If you're prompted to install the NuGet provider, type Y and press ENTER.
6. If you're prompted to install the module from [PSGallery](#), type Y and press ENTER.

Install support for PSCredential

Azure Automation uses the [PSCredential](#) class to represent a credential asset. Your scripts retrieve [PSCredential](#) objects using the [Get-AutomationPSCredential](#) cmdlet. For more information, see [Credential assets in Azure Automation](#).

Assign a subscription administrator

You must assign an administrator for the Azure subscription. This person has the role of Owner for the subscription scope. See [Role-based access control in Azure Automation](#).

Change the Azure AD user's password

To change the Azure AD user's password:

1. Log out of Azure.
2. Have the administrator log in to Azure as the Azure AD user just created, using the full user name (including the domain) and a temporary password.
3. Ask the administrator to change the password when prompted.

Configure Azure Automation to manage the Azure subscription

For Azure Automation to communicate with Azure AD, you must retrieve the credentials associated with the Azure connection to Azure AD. Examples of these credentials are tenant ID, subscription ID, and the like. For more about the connection between Azure and Azure AD, see [Connect your organization to Azure Active Directory](#).

Create a credential asset

With the Azure credentials for Azure AD available, it's time to create an Azure Automation credential asset to securely store the Azure AD credentials so that runbooks and Desired State Configuration (DSC) scripts can access them. You can do this using either the Azure portal or PowerShell cmdlets.

Create the credential asset in Azure portal

You can use the Azure portal to create the credential asset. Do this operation from your Automation account using [Credentials](#) under [Shared Resources](#). See [Credential assets in Azure Automation](#).

Create the credential asset with Windows PowerShell

To prepare a new credential asset in Windows PowerShell, your script first creates a [PSCredential](#) object using the assigned user name and password. The script then uses this object to create the asset through a call to the [New-AzureAutomationCredential](#) cmdlet. Alternatively, the script can call the [Get-Credential](#) cmdlet to prompt the user to type in a name and password. See [Credential assets in Azure Automation](#).

Manage Azure resources from an Azure Automation runbook

You can manage Azure resources from Azure Automation runbooks using the credential asset. Below is an example PowerShell runbook that collects the credential asset to use for stopping and starting virtual machines in an Azure subscription. This runbook first uses `Get-AutomationPSCredential` to retrieve the credential to use to authenticate to Azure. It then calls the `Connect-AzAccount` cmdlet to connect to Azure using the credential. The script uses the `Select-AzureSubscription` cmdlet to choose the subscription to work with.

```
Workflow Stop-Start-AzureVM
{
    Param
    (
        [Parameter(Mandatory=$true)][ValidateNotNullOrEmpty()]
        [String]
        $AzureSubscriptionId,
        [Parameter(Mandatory=$true)][ValidateNotNullOrEmpty()]
        [String]
        $AzureVMList="All",
        [Parameter(Mandatory=$true)][ValidateSet("Start","Stop")]
        [String]
        $Action
    )

    $credential = Get-AutomationPSCredential -Name 'AzureCredential'
    Connect-AzAccount -Credential $credential
    Select-AzureSubscription -SubscriptionId $AzureSubscriptionId

    if($AzureVMList -ne "All")
    {
        $AzureVMs = $AzureVMList.Split(",")
        [System.Collections.ArrayList]$AzureVMsToHandle = $AzureVMs
    }
    else
    {
        $AzureVMs = (Get-AzVM).Name
        [System.Collections.ArrayList]$AzureVMsToHandle = $AzureVMs
    }

    foreach($AzureVM in $AzureVMsToHandle)
    {
        if(!(Get-AzVM | ? {$_ .Name -eq $AzureVM}))
        {
            throw " AzureVM : [$AzureVM] - Does not exist! - Check your inputs "
        }
    }

    if($Action -eq "Stop")
    {
        Write-Output "Stopping VMs";
        foreach -parallel ($AzureVM in $AzureVMsToHandle)
        {
            Get-AzVM | ? {$_ .Name -eq $AzureVM} | Stop-AzVM -Force
        }
    }
    else
    {
        Write-Output "Starting VMs";
        foreach -parallel ($AzureVM in $AzureVMsToHandle)
        {
            Get-AzVM | ? {$_ .Name -eq $AzureVM} | Start-AzVM
        }
    }
}
```

Next steps

- For details of credential use, see [Manage credentials in Azure Automation](#).
- For information about modules, see [Manage modules in Azure Automation](#).
- If you need to start a runbook, see [Start a runbook in Azure Automation](#).
- For PowerShell details, see [PowerShell Docs](#).

DNS records for Azure regions used by Azure Automation

7/8/2021 • 4 minutes to read • [Edit Online](#)

The [Azure Automation](#) service uses a number of DNS records for features to connect to the service. If you have an Automation account that's defined for a specific region, you can restrict communication to that regional datacenter. You might need to know these records to allow the following Automation features to work when it is hosted behind a firewall:

- Hybrid Runbook Worker
- State Configuration
- Webhooks

NOTE

Linux Hybrid Runbook Worker registration will fail with the new records unless it is version 1.6.10.2 or higher. You must upgrade to a newer version of the [Log Analytics agent for Linux](#) in order for the machine to receive an updated version of the worker role and use these new records. Existing machines will continue working without any issues.

DNS records per region

The following table provides the DNS record for each region.

NOTE

While the list of Automation DNS records provided here have been retired, they still remain functional to allow time for you to migrate to the new records listed under [support for Private Link](#) and prevent failures with your automation processes.

REGION	DNS RECORD
Australia Central	ac-jobruntimedata-prod-su1.azure-automation.net ac-agentservice-prod-1.azure-automation.net
Australia East	ae-jobruntimedata-prod-su1.azure-automation.net ae-agentservice-prod-1.azure-automation.net
Australia South East	ase-jobruntimedata-prod-su1.azure-automation.net ase-agentservice-prod-1.azure-automation.net
Canada Central	cc-jobruntimedata-prod-su1.azure-automation.net cc-agentservice-prod-1.azure-automation.net
Central India	cid-jobruntimedata-prod-su1.azure-automation.net cid-agentservice-prod-1.azure-automation.net
East US 2	eus2-jobruntimedata-prod-su1.azure-automation.net eus2-agentservice-prod-1.azure-automation.net

REGION	DNS RECORD
Japan East	jpe-jobruntimedata-prod-su1.azure-automation.net jpe-agentservice-prod-1.azure-automation.net
North Europe	ne-jobruntimedata-prod-su1.azure-automation.net ne-agentservice-prod-1.azure-automation.net
South Central US	scus-jobruntimedata-prod-su1.azure-automation.net scus-agentservice-prod-1.azure-automation.net
South East Asia	sea-jobruntimedata-prod-su1.azure-automation.net sea-agentservice-prod-1.azure-automation.net
Switzerland North	stzn-jobruntimedata-prod-su1.azure-automation.net stzn-agentservice-prod-su1.azure-automation.net
UK South	uks-jobruntimedata-prod-su1.azure-automation.net uks-agentservice-prod-1.azure-automation.net
US Gov Virginia	usge-jobruntimedata-prod-su1.azure-automation.us usge-agentservice-prod-1.azure-automation.us
West Central US	wcus-jobruntimedata-prod-su1.azure-automation.net wcus-agentservice-prod-1.azure-automation.net
West Europe	we-jobruntimedata-prod-su1.azure-automation.net we-agentservice-prod-1.azure-automation.net
West US 2	wus2-jobruntimedata-prod-su1.azure-automation.net wus2-agentservice-prod-1.azure-automation.net

Support for Private Link

To support [Private Link](#) in Azure Automation, the DNS records for every supported datacenter have been updated. Instead of region-specific URLs, the URLs are Automation account specific.

REGION	DNS RECORD
South Africa North	<code>https://<accountId>.webhook.san.azure-automation.net</code> <code>https://<accountId>.agentsvc.san.azure-automation.net</code> <code>https://<accountId>.jrds.san.azure-automation.net</code>
East Asia	<code>https://<accountId>.webhook.ea.azure-automation.net</code> <code>https://<accountId>.agentsvc.ea.azure-automation.net</code> <code>https://<accountId>.jrds.ea.azure-automation.net</code>
South East Asia	<code>https://<accountId>.webhook.sea.azure-automation.net</code> <code>https://<accountId>.agentsvc.sea.azure-automation.net</code> <code>https://<accountId>.jrds.sea.azure-automation.net</code>

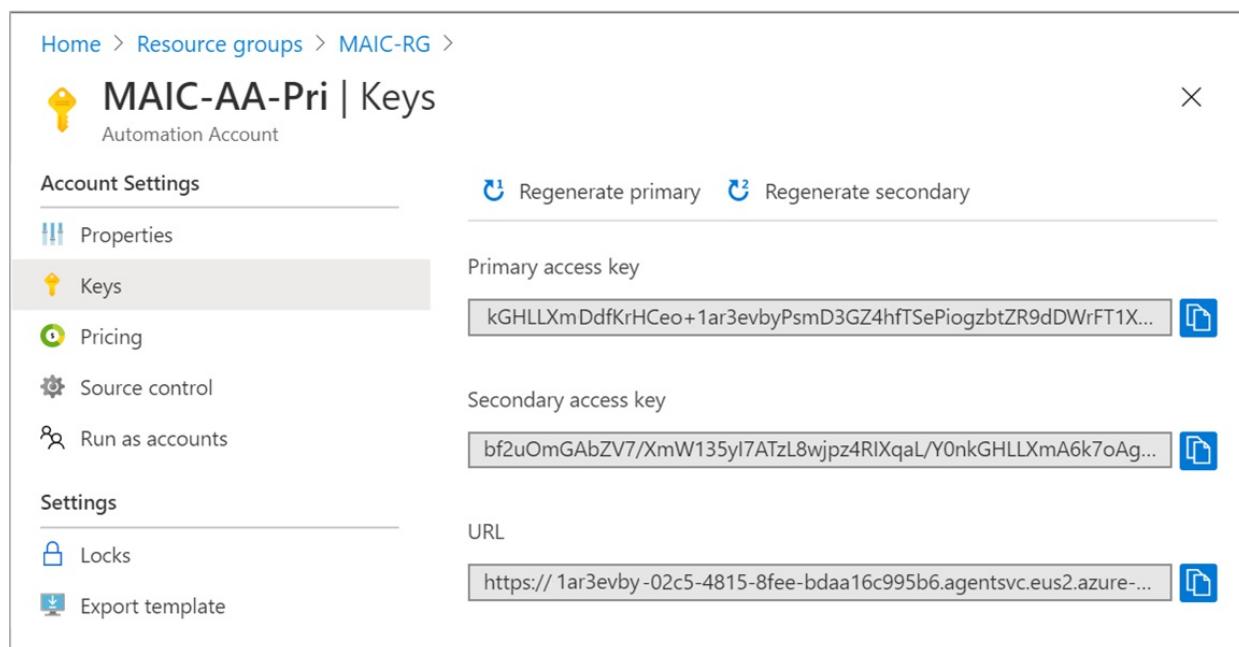
REGION	DNS RECORD
Australia Central	<pre>https://<accountId>.webhook.ac.azure-automation.net https://<accountId>.agentsvc.ac.azure- automation.net https://<accountId>.jrds.ac.azure-automation.net</pre>
Australia Central 2	<pre>https://<accountId>.webhook.cbr2.azure- automation.net https://<accountId>.agentsvc.cbr2.azure- automation.net https://<accountId>.jrds.cbr2.azure-automation.net</pre>
Australia South East	<pre>https://<accountId>.webhook.ase.azure- automation.net https://<accountId>.agentsvc.ase.azure- automation.net https://<accountId>.jrds.ase.azure-automation.net</pre>
Australia East	<pre>https://<accountId>.webhook.ae.azure-automation.net https://<accountId>.agentsvc.ae.azure- automation.net https://<accountId>.jrds.ae.azure-automation.net</pre>
Brazil South	<pre>https://<accountId>.webhook.brs.azure- automation.net https://<accountId>.agentsvc.brs.azure- automation.net https://<accountId>.jrds.brs.azure-automation.net</pre>
Brazil Southeast	<pre>https://<accountId>.webhook.brse.azure- automation.net https://<accountId>.agentsvc.brse.azure- automation.net https://<accountId>.jrds.brse.azure-automation.net</pre>
Canada Central	<pre>https://<accountId>.webhook.cc.azure-automation.net https://<accountId>.agentsvc.cc.azure- automation.net https://<accountId>.jrds.cc.azure-automation.net</pre>
China East 2	<pre>https://<accountId>.webhook.sha2.azure- automation.cn https://<accountId>.agentsvc.sha2.azure- automation.cn https://<accountId>.jrds.sha2.azure-automation.cn</pre>
China North	<pre>https://<accountId>.webhook.bjb.azure-automation.cn https://<accountId>.agentsvc.bjb.azure- automation.cn https://<accountId>.jrds.bjb.azure-automation.cn</pre>
China North 2	<pre>https://<accountId>.webhook.bjs2.azure- automation.cn https://<accountId>.agentsvc.bjs2.azure- automation.cn https://<accountId>.jrds.bjs2.azure-automation.cn</pre>

REGION	DNS RECORD
West Europe	<pre>https://<accountId>.webhook.we.azure-automation.net https://<accountId>.agentsvc.we.azure-automation.net https://<accountId>.jrds.we.azure-automation.net</pre>
North Europe	<pre>https://<accountId>.webhook.ne.azure-automation.net https://<accountId>.agentsvc.ne.azure-automation.net https://<accountId>.jrds.ne.azure-automation.net</pre>
France Central	<pre>https://<accountId>.webhook.fc.azure-automation.net https://<accountId>.agentsvc.fc.azure-automation.net https://<accountId>.jrds.fc.azure-automation.net</pre>
France South	<pre>https://<accountId>.webhook.mrs.azure-automation.net https://<accountId>.agentsvc.mrs.azure-automation.net https://<accountId>.jrds.mrs.azure-automation.net</pre>
Germany West Central	<pre>https://<accountId>.webhook.dewc.azure-automation.de https://<accountId>.agentsvc.dewc.azure-automation.de https://<accountId>.jrds.dewc.azure-automation.de</pre>
Central India	<pre>https://<accountId>.webhook.cid.azure-automation.net https://<accountId>.agentsvc.cid.azure-automation.net https://<accountId>.jrds.cid.azure-automation.net</pre>
South India	<pre>https://<accountId>.webhook.ma.azure-automation.net https://<accountId>.agentsvc.ma.azure-automation.net https://<accountId>.jrds.ma.azure-automation.net</pre>
Japan East	<pre>https://<accountId>.webhook.jpe.azure-automation.net https://<accountId>.agentsvc.jpe.azure-automation.net https://<accountId>.jrds.jpe.azure-automation.net</pre>
Japan West	<pre>https://<accountId>.webhook.jpw.azure-automation.net https://<accountId>.agentsvc.jpw.azure-automation.net https://<accountId>.jrds.jpw.azure-automation.net</pre>
Korea Central	<pre>https://<accountId>.webhook.kc.azure-automation.net https://<accountId>.agentsvc.kc.azure-automation.net https://<accountId>.jrds.kc.azure-automation.net</pre>

REGION	DNS RECORD
Korea South	<pre>https://<accountId>.webhook.ps.azure-automation.net https://<accountId>.agentsvc.ps.azure-automation.net https://<accountId>.jrds.ps.azure-automation.net</pre>
Norway East	<pre>https://<accountId>.webhook.noe.azure-automation.net https://<accountId>.agentsvc.noe.azure-automation.net https://<accountId>.jrds.noe.azure-automation.net</pre>
Norway West	<pre>https://<accountId>.webhook.now.azure-automation.net https://<accountId>.agentsvc.now.azure-automation.net https://<accountId>.jrds.now.azure-automation.net</pre>
Switzerland West	<pre>https://<accountId>.webhook.stzw.azure-automation.net https://<accountId>.agentsvc.stzw.azure-automation.net https://<accountId>.jrds.stzw.azure-automation.net</pre>
UAE Central	<pre>https://<accountId>.webhook.auh.azure-automation.net https://<accountId>.agentsvc.auh.azure-automation.net https://<accountId>.jrds.auh.azure-automation.net</pre>
UAE North	<pre>https://<accountId>.webhook.uaen.azure-automation.net https://<accountId>.agentsvc.uaen.azure-automation.net https://<accountId>.jrds.uaen.azure-automation.net</pre>
UK West	<pre>https://<accountId>.webhook.cw.azure-automation.net https://<accountId>.agentsvc.cw.azure-automation.net https://<accountId>.jrds.cw.azure-automation.net</pre>
UK South	<pre>https://<accountId>.webhook.uks.azure-automation.net https://<accountId>.agentsvc.uks.azure-automation.net https://<accountId>.jrds.uks.azure-automation.net</pre>
Central US	<pre>https://<accountId>.webhook.cus.azure-automation.net https://<accountId>.agentsvc.cus.azure-automation.net https://<accountId>.jrds.cus.azure-automation.net</pre>
East US	<pre>https://<accountId>.webhook.eus.azure-automation.net https://<accountId>.agentsvc.eus.azure-automation.net https://<accountId>.jrds.eus.azure-automation.net</pre>

REGION	DNS RECORD
East US 2	<pre>https://<accountId>.webhook.eus2.azure-automation.net</pre> <pre>https://<accountId>.agentsvc.eus2.azure-automation.net</pre> <pre>https://<accountId>.jrds.eus2.azure-automation.net</pre>
North Central US	<pre>https://<accountId>.webhook.ncus.azure-automation.net</pre> <pre>https://<accountId>.agentsvc.ncus.azure-automation.net</pre> <pre>https://<accountId>.jrds.ncus.azure-automation.net</pre>
South Central US	<pre>https://<accountId>.webhook.scus.azure-automation.net</pre> <pre>https://<accountId>.agentsvc.scus.azure-automation.net</pre> <pre>https://<accountId>.jrds.scus.azure-automation.net</pre>
West Central US	<pre>https://<accountId>.webhook.wcus.azure-automation.net</pre> <pre>https://<accountId>.agentsvc.wcus.azure-automation.net</pre> <pre>https://<accountId>.jrds.wcus.azure-automation.net</pre>
West US	<pre>https://<accountId>.webhook.wus.azure-automation.net</pre> <pre>https://<accountId>.agentsvc.wus.azure-automation.net</pre> <pre>https://<accountId>.jrds.wus.azure-automation.net</pre>
West US 2	<pre>https://<accountId>.webhook.wus2.azure-automation.net</pre> <pre>https://<accountId>.agentsvc.wus2.azure-automation.net</pre> <pre>https://<accountId>.jrds.wus2.azure-automation.net</pre>
West US 3	<pre>https://<accountId>.webhook.usw3.azure-automation.net</pre> <pre>https://<accountId>.agentsvc.usw3.azure-automation.net</pre> <pre>https://<accountId>.jrds.usw3.azure-automation.net</pre>
US Gov Virginia	<pre>https://<accountId>.webhook.usge.azure-automation.us</pre> <pre>https://<accountId>.agentsvc.usge.azure-automation.us</pre> <pre>https://<accountId>.jrds.usge.azure-automation.us</pre>
US Gov Texas	<pre>https://<accountId>.webhook.ussc.azure-automation.us</pre> <pre>https://<accountId>.agentsvc.ussc.azure-automation.us</pre> <pre>https://<accountId>.jrds.ussc.azure-automation.us</pre>
US Gov Arizona	<pre>https://<accountId>.webhook.phx.azure-automation.us</pre> <pre>https://<accountId>.agentsvc.phx.azure-automation.us</pre> <pre>https://<accountId>.jrds.phx.azure-automation.us</pre>

Replace <accountID> in the DNS record with GUID representing your Automation Account ID from the value URL. You can get the ID required from Keys under Account Settings in the Azure portal.



Copy the value after *accounts/* from the URL field -

<https://<GUID>.azuresvc.<region>.azure-automation.net/accounts/<GUID>>

NOTE

All of the Webhook and agentservice DNS records have been updated to the new style DNS records to support Private Link. For JRDS DNS records, both old and new style DNS records are supported. If you are not using Private Link, you will see the old style DNS records, while those using Private Link will see new style of DNS records.

We recommend that you use the addresses listed when defining [exceptions](#). For a list of region IP addresses instead of region names, download the JSON file from the Microsoft Download Center for the following cloud environments:

- [Azure IP Ranges and Service Tags - Azure public](#)
- [Azure IP Ranges and Service Tags- Azure Government](#)
- [Azure IP Ranges and Service Tags - Azure Germany](#)
- [Azure IP Ranges and Service Tags - Azure China Vianet 21](#)

The IP address file lists the IP address ranges that are used in the Microsoft Azure datacenters. It includes compute, SQL, and storage ranges, and reflects currently deployed ranges and any upcoming changes to the IP ranges. New ranges that appear in the file aren't used in the datacenters for at least one week.

It's a good idea to download the new IP address file every week. Then, update your site to correctly identify services running in Azure.

NOTE

If you're using Azure ExpressRoute, remember that the IP address file is used to update the Border Gateway Protocol (BGP) advertisement of Azure space in the first week of each month.

Next steps

- To learn how to troubleshoot your Hybrid Runbook Workers, see [Troubleshoot Hybrid Runbook Worker](#)

issues.

- To learn how to troubleshoot issues with State Configuration, see [Troubleshoot State Configuration issues](#).

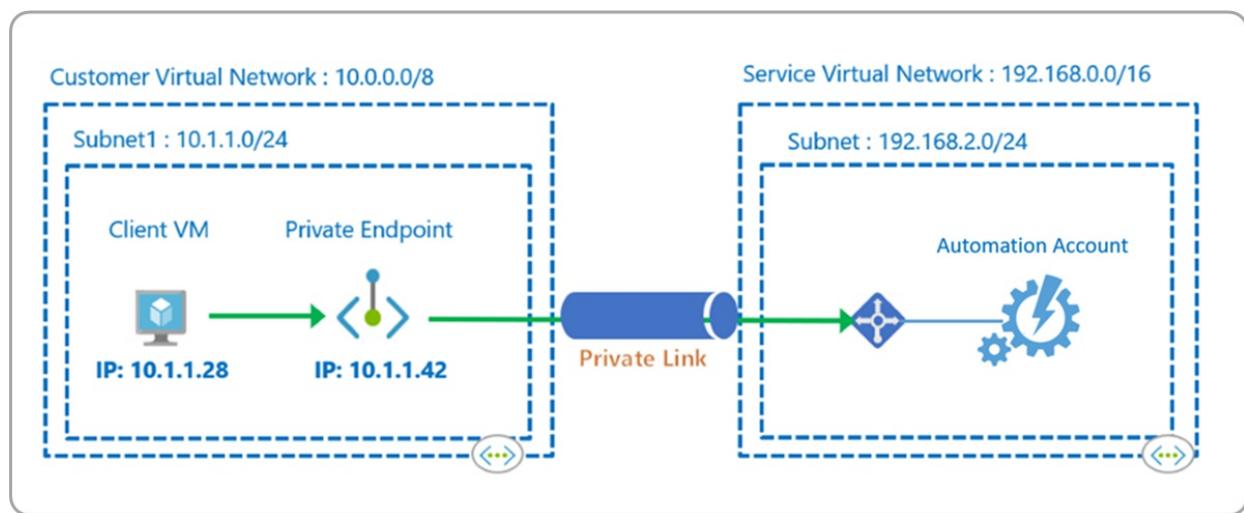
Use Azure Private Link to securely connect networks to Azure Automation

8/24/2021 • 9 minutes to read • [Edit Online](#)

Azure Private Endpoint is a network interface that connects you privately and securely to a service powered by Azure Private Link. Private Endpoint uses a private IP address from your VNet, effectively bringing the Automation service into your VNet. Network traffic between the machines on the VNet and the Automation account traverses over the VNet and a private link on the Microsoft backbone network, eliminating exposure from the public internet.

For example, you have a VNet where you have disabled outbound internet access. However, you want to access your Automation account privately and use Automation features like Webhooks, State Configuration, and runbook jobs on Hybrid Runbook Workers. Moreover, you want users to have access to the Automation account only through the VNET. Deploying private endpoint achieves these goals.

This article covers when to use and how to set up a private endpoint with your Automation account.



NOTE

Private Link support with Azure Automation is available only in Azure Commercial and Azure US Government clouds.

Advantages

With Private Link you can:

- Connect privately to Azure Automation without opening up any public network access.
- Connect privately to Azure Monitor Log Analytics workspace without opening any public network access.

NOTE

A separate private endpoint for your Log Analytics workspace is required if your Automation account is linked to a Log Analytics workspace to forward job data, and when you have enabled features such as Update Management, Change Tracking and Inventory, State Configuration, or Start/Stop VMs during off-hours. For more information about Private Link for Azure Monitor, see [Use Azure Private Link to securely connect networks to Azure Monitor](#).

- Ensure your Automation data is only accessed through authorized private networks.
- Prevent data exfiltration from your private networks by defining your Azure Automation resource that connects through your private endpoint.
- Securely connect your private on-premises network to Azure Automation using ExpressRoute and Private Link.
- Keep all traffic inside the Microsoft Azure backbone network.

For more information, see [Key Benefits of Private Link](#).

Limitations

- In the current implementation of Private Link, Automation account cloud jobs cannot access Azure resources that are secured using private endpoint. For example, Azure Key Vault, Azure SQL, Azure Storage account, etc. To workaround this, use a [Hybrid Runbook Worker](#) instead.
- You need to use the latest version of the [Log Analytics agent](#) for Windows or Linux.
- The [Log Analytics Gateway](#) does not support Private Link.

How it works

Azure Automation Private Link connects one or more private endpoints (and therefore the virtual networks they are contained in) to your Automation Account resource. These endpoints are machines using webhooks to start a runbook, machines hosting the Hybrid Runbook Worker role, and Desired State Configuration (DSC) nodes.

After you create private endpoints for Automation, each of the public facing Automation URLs, is mapped to one private endpoint in your VNet. You or a machine can directly contact the Automation URLs.

Webhook scenario

You can start runbooks by doing a POST on the webhook URL. For example, the URL looks like:

```
https://<automationAccountId>.webhooks.<region>.azure-automation.net/webhooks?
token=gzGMz4SMpqNo8gidqPxAJ3E%3d
```

Hybrid Runbook Worker scenario

The user Hybrid Runbook Worker feature of Azure Automation enables you to run runbooks directly on the Azure or non-Azure machine, including servers registered with Azure Arc-enabled servers. From the machine or server that's hosting the role, you can run runbooks directly on it and against resources in the environment to manage those local resources.

A JRDS endpoint is used by the hybrid worker to start/stop runbooks, download the runbooks to the worker, and to send the job log stream back to the Automation service. After enabling JRDS endpoint, the URL would look like this: <https://<automationaccountID>.jrds.<region>.privatelink.azure-automation.net>. This would ensure runbook execution on the hybrid worker connected to Azure Virtual Network is able to execute jobs without the need to open an outbound connection to the Internet.

NOTE

With the current implementation of Private Links for Azure Automation, it only supports running jobs on the Hybrid Runbook Worker connected to an Azure virtual network and does not support cloud jobs.

Hybrid Worker scenario for Update Management

The system Hybrid Runbook Worker supports a set of hidden runbooks used by the Update Management feature that are designed to install user-specified updates on Windows and Linux machines. When Azure

Automation Update Management is enabled, any machine connected to your Log Analytics workspace is automatically configured as a system Hybrid Runbook Worker.

To understand & configure Update Management review [About Update Management](#). The Update Management feature has a dependency on a Log Analytics workspace, and therefore requires linking the workspace with an Automation account. A Log Analytics workspace stores data collected by the solution, and host its log searches and views.

If you want your machines configured for Update management to connect to Automation & Log Analytics workspace in a secure manner over Private Link channel, you have to enable Private Link for the Log Analytics workspace linked to the Automation Account configured with Private Link.

You can control how a Log Analytics workspace can be reached from outside of the Private Link scopes by following the steps described in [Configure Log Analytics](#). If you set **Allow public network access for ingestion** to **No**, then machines outside of the connected scopes cannot upload data to this workspace. If you set **Allow public network access for queries** to **No**, then machines outside of the scopes cannot access data in this workspace.

Use **DSCAndHybridWorker** target sub-resource to enable Private Link for user & system hybrid workers.

NOTE

Machines hosted outside of Azure that are managed by Update Management and are connected to the Azure VNet over ExpressRoute private peering, VPN tunnels, and peered virtual networks using private endpoints support Private Link.

State Configuration (agentsvc) scenario

State Configuration provides you with Azure configuration management service that allows you to write, manage, and compile PowerShell Desired State Configuration (DSC) configurations for nodes in any cloud or on-premises datacenter.

The agent on the machine registers with DSC service and then uses the service endpoint to pull DSC configuration. The agent service endpoint looks like:

`https://<automationAccountId>.agentsvc.<region>.azure-automation.net .`

The URL for public & private endpoint would be the same, however, it would be mapped to a private IP address when Private link is enabled.

Planning based on your network

Before setting up your Automation account resource, consider your network isolation requirements. Evaluate your virtual networks' access to public internet, and the access restrictions to your Automation account (including setting up a Private Link Group Scope to Azure Monitor Logs if integrated with your Automation account). Also include a review of the Automation service [DNS records](#) as part of your plan to ensure the supported features work without issue.

Connect to a private endpoint

Create a private endpoint to connect our network. You can create it in the [Azure portal Private Link center](#). Once your changes to publicNetworkAccess and Private Link are applied, it can take up to 35 minutes for them to take effect.

In this section, you'll create a private endpoint for your Automation account.

1. On the upper-left side of the screen, select **Create a resource > Networking > Private Link Center**.
2. In **Private Link Center - Overview**, on the option to **Build a private connection to a service**, select **Start**.

3. In **Create a virtual machine - Basics**, enter or select the following information:

SETTING	VALUE
PROJECT DETAILS	
Subscription	Select your subscription.
Resource group	Select myResourceGroup . You created this in the previous section.
INSTANCE DETAILS	
Name	Enter your <i>PrivateEndpoint</i> .
Region	Select YourRegion .

4. Select **Next: Resource**.

5. In **Create a private endpoint - Resource**, enter or select the following information:

SETTING	VALUE
Connection method	Select connect to an Azure resource in my directory.
Subscription	Select your subscription.
Resource type	Select Microsoft.Automation/automationAccounts .
Resource	Select <i>myAutomationAccount</i>
Target subresource	Select <i>Webhook</i> or <i>DSCAndHybridWorker</i> depending on your scenario.

6. Select **Next: Configuration**.

7. In **Create a private endpoint - Configuration**, enter or select the following information:

SETTING	VALUE
NETWORKING	
Virtual network	Select <i>MyVirtualNetwork</i> .
Subnet	Select <i>mySubnet</i> .
PRIVATE DNS INTEGRATION	
Integrate with private DNS zone	Select Yes .
Private DNS Zone	Select <i>(New)privatelink.azure-automation.net</i>

SETTING	VALUE

8. Select **Review + create**. You're taken to the **Review + create** page where Azure validates your configuration.

9. When you see the **Validation passed** message, select **Create**.

In the **Private Link Center**, select **Private endpoints** to view your private link resource.

Name	Resource	Target sub-resource	Subnet	Connection state
my-automation-private-link	my-automation-acc	Webhook	Approved	Approved

Select the resource to see all the details. This creates a new private endpoint for your Automation account and assigns it a private IP from your virtual network. The **Connection status** shows as **approved**.

Similarly, a unique fully qualified domain name (FQDN) is created for the State Configuration (agentsvc) and for the Hybrid Runbook Worker job runtime (jrds). Each of them are assigned a separate IP from your VNet and the **Connection status** shows as **approved**.

If the service consumer has Azure RBAC permissions on the Automation resource, they can choose the automatic approval method. In this case, when the request reaches the Automation provider resource, no action is required from the service provider and the connection is automatically approved.

Set public network access flags

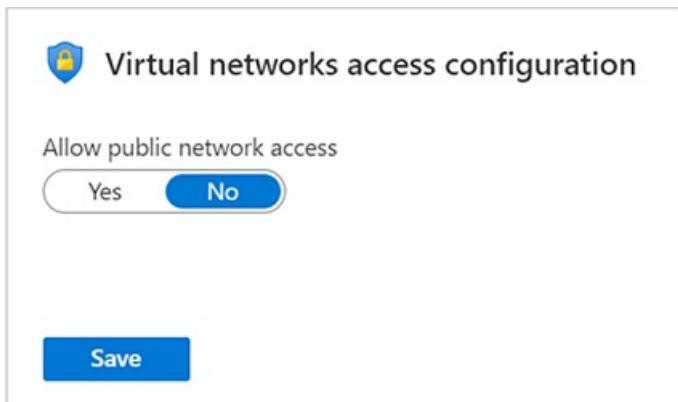
You can configure an Automation account to deny all public configurations and allow only connections through private endpoints to further enhance the network security. If you want to restrict access to the Automation account from only within the VNet and not allow access from public internet, you can set `publicNetworkAccess` property to `$false`.

When the **Public Network Access** setting is set to `$false`, only connections via private endpoints are allowed and all connections via public endpoints are denied with an unauthorized error message and HTTP status of 401.

The following PowerShell script shows how to `Get` and `Set` the **Public Network Access** property at the Automation account level:

```
$account = Get-AzResource -ResourceType Microsoft.Automation/automationAccounts -ResourceGroupName "<resourceGroupName>" -Name "<automationAccountName>" -ApiVersion "2020-01-13-preview"
$account.Properties | Add-Member -Name 'publicNetworkAccess' -Type NoteProperty -Value $false
$account | Set-AzResource -Force -ApiVersion "2020-01-13-preview"
```

You can also control the public network access property from the Azure portal. From your Automation Account, select **Network Isolation** from the left-hand pane under the **Account Settings** section. When the Public Network Access setting is set to **No**, only connections over private endpoints are allowed and all connections over public endpoints are denied.



DNS configuration

When connecting to a private link resource using a fully qualified domain name (FQDN) as part of the connection string, it's important to correctly configure your DNS settings to resolve to the allocated private IP address. Existing Azure services might already have a DNS configuration to use when connecting over a public endpoint. Your DNS configuration should be reviewed and updated to connect using your private endpoint.

The network interface associated with the private endpoint contains the complete set of information required to configure your DNS, including FQDN and private IP addresses allocated for a given private link resource.

You can use the following options to configure your DNS settings for private endpoints:

- Use the host file (only recommended for testing). You can use the host file on a virtual machine to override using DNS for name resolution first. Your DNS entry should look like the following example:
`privatelinkFQDN.jrds.sea.azure-automation.net .`
- Use a **private DNS zone**. You can use private DNS zones to override the DNS resolution for a particular private endpoint. A private DNS zone can be linked to your virtual network to resolve specific domains. To enable the agent on your virtual machine to communicate over the private endpoint, create a Private DNS record as `privatelink.azure-automation.net`. Add a new DNS A record mapping to the IP of the private endpoint.
- Use your DNS forwarder (optional). You can use your DNS forwarder to override the DNS resolution for a particular private link resource. If your **DNS server** is hosted on a virtual network, you can create a DNS forwarding rule to use a private DNS zone to simplify the configuration for all private link resources.

For more information, see [Azure Private Endpoint DNS configuration](#).

Next steps

To learn more about Private Endpoint, see [What is Azure Private Endpoint?](#).

Manage role permissions and security

8/30/2021 • 18 minutes to read • [Edit Online](#)

Azure role-based access control (Azure RBAC) enables access management for Azure resources. Using [Azure RBAC](#), you can segregate duties within your team and grant only the amount of access to users, groups, and applications that they need to perform their jobs. You can grant role-based access to users using the Azure portal, Azure Command-Line tools, or Azure Management APIs.

Roles in Automation accounts

In Azure Automation, access is granted by assigning the appropriate Azure role to users, groups, and applications at the Automation account scope. Following are the built-in roles supported by an Automation account:

ROLE	DESCRIPTION
Owner	The Owner role allows access to all resources and actions within an Automation account including providing access to other users, groups, and applications to manage the Automation account.
Contributor	The Contributor role allows you to manage everything except modifying other user's access permissions to an Automation account.
Reader	The Reader role allows you to view all the resources in an Automation account but can't make any changes.
Automation Contributor	The Automation Contributor role allows you to manage all resources in the Automation account, except modifying other user's access permissions to an Automation account.
Automation Operator	The Automation Operator role allows you to view runbook name and properties and to create and manage jobs for all runbooks in an Automation account. This role is helpful if you want to protect your Automation account resources like credentials assets and runbooks from being viewed or modified but still allow members of your organization to execute these runbooks.
Automation Job Operator	The Automation Job Operator role allows you to create and manage jobs for all runbooks in an Automation account.
Automation Runbook Operator	The Automation Runbook Operator role allows you to view a runbook's name and properties.

ROLE	DESCRIPTION
Log Analytics Contributor	The Log Analytics Contributor role allows you to read all monitoring data and edit monitoring settings. Editing monitoring settings includes adding the VM extension to VMs, reading storage account keys to be able to configure collection of logs from Azure storage, creating and configuring Automation accounts, adding Azure Automation features, and configuring Azure diagnostics on all Azure resources.
Log Analytics Reader	The Log Analytics Reader role allows you to view and search all monitoring data as well as view monitoring settings. This includes viewing the configuration of Azure diagnostics on all Azure resources.
Monitoring Contributor	The Monitoring Contributor role allows you to read all monitoring data and update monitoring settings.
Monitoring Reader	The Monitoring Reader role allows you to read all monitoring data.
User Access Administrator	The User Access Administrator role allows you to manage user access to Azure Automation accounts.

Role permissions

The following tables describe the specific permissions given to each role. This can include Actions, which give permissions, and NotActions, which restrict them.

Owner

An Owner can manage everything, including access. The following table shows the permissions granted for the role:

ACTIONS	DESCRIPTION
Microsoft.Automation/automationAccounts/	Create and manage resources of all types.

Contributor

A Contributor can manage everything except access. The following table shows the permissions granted and denied for the role:

ACTIONS	DESCRIPTION
Microsoft.Automation/automationAccounts/	Create and manage resources of all types
Not Actions	
Microsoft.Authorization/*/Delete	Delete roles and role assignments.
Microsoft.Authorization/*/Write	Create roles and role assignments.
Microsoft.Authorization/elevateAccess/Action	Denies the ability to create a User Access Administrator.

Reader

A Reader can view all the resources in an Automation account but can't make any changes.

ACTIONS	DESCRIPTION
Microsoft.Automation/automationAccounts/read	View all resources in an Automation account.

Automation Contributor

An Automation Contributor can manage all resources in the Automation account except access. The following table shows the permissions granted for the role:

ACTIONS	DESCRIPTION
Microsoft.Automation/automationAccounts/*	Create and manage resources of all types under Automation account.
Microsoft.Authorization/*/read	Read roles and role assignments.
Microsoft.Resources/deployments/*	Create and manage resource group deployments.
Microsoft.Resources/subscriptions/resourceGroups/read	Read resource group deployments.
Microsoft.Support/*	Create and manage support tickets.

NOTE

The Automation Contributor role can be used to access any resource using the managed identity, if appropriate permissions are set on the target resource, or using a Run As account. An Automation Run As account are by default, configured with Contributor rights on the subscription. Follow the principle of least privilege and carefully assign permissions only required to execute your runbook. For example, if the Automation account is only required to start or stop an Azure VM, then the permissions assigned to the Run As account or managed identity needs to be only for starting or stopping the VM. Similarly, if a runbook is reading from blob storage, then assign read only permissions.

When assigning permissions, it is recommended to use Azure role based access control (RBAC) assigned to a managed identity. Review our [best approach](#) recommendations for using a system or user-assigned managed identity, including management and governance during its lifetime.

Automation Operator

An Automation Operator is able to create and manage jobs, and read runbook names and properties for all runbooks in an Automation account.

NOTE

If you want to control operator access to individual runbooks then don't set this role. Instead use the **Automation Job Operator** and **Automation Runbook Operator** roles in combination.

The following table shows the permissions granted for the role:

ACTIONS	DESCRIPTION
Microsoft.Authorization/*/read	Read authorization.
Microsoft.Automation/automationAccounts/hybridRunbookWorkerGroups/read	Read Hybrid Runbook Worker Resources.

ACTIONS	DESCRIPTION
Microsoft.Automation/automationAccounts/jobs/read	List jobs of the runbook.
Microsoft.Automation/automationAccounts/jobs/resume/action	Resume a job that is paused.
Microsoft.Automation/automationAccounts/jobs/stop/action	Cancel a job in progress.
Microsoft.Automation/automationAccounts/jobsstreams/read	Read the Job Streams and Output.
Microsoft.Automation/automationAccounts/jobs/output/read	Get the Output of a job.
Microsoft.Automation/automationAccounts/jobs/suspend/action	Pause a job in progress.
Microsoft.Automation/automationAccounts/jobs/write	Create jobs.
Microsoft.Automation/automationAccounts/jobSchedules/read	Get an Azure Automation job schedule.
Microsoft.Automation/automationAccounts/jobSchedules/write	Create an Azure Automation job schedule.
Microsoft.Automation/automationAccounts/linkedWorkspace/read	Get the workspace linked to the Automation account.
Microsoft.Automation/automationAccounts/read	Get an Azure Automation account.
Microsoft.Automation/automationAccounts/runbooks/read	Get an Azure Automation runbook.
Microsoft.Automation/automationAccounts/schedules/read	Get an Azure Automation schedule asset.
Microsoft.Automation/automationAccounts/schedules/write	Create or update an Azure Automation schedule asset.
Microsoft.Resources/subscriptions/resourceGroups/read	Read roles and role assignments.
Microsoft.Resources/deployments/*	Create and manage resource group deployments.
Microsoft.Insights/alertRules/*	Create and manage alert rules.
Microsoft.Support/*	Create and manage support tickets.

Automation Job Operator

An Automation Job Operator role is granted at the Automation account scope. This allows the operator permissions to create and manage jobs for all runbooks in the account. If the Job Operator role is granted read permissions on the resource group containing the Automation account, members of the role have the ability to start runbooks. However, they don't have the ability to create, edit, or delete them.

The following table shows the permissions granted for the role:

ACTIONS	DESCRIPTION
Microsoft.Authorization/*/read	Read authorization.
Microsoft.Automation/automationAccounts/jobs/read	List jobs of the runbook.
Microsoft.Automation/automationAccounts/jobs/resume/action	Resume a job that is paused.
Microsoft.Automation/automationAccounts/jobs/stop/action	Cancel a job in progress.
Microsoft.Automation/automationAccounts/jobsstreams/read	Read the Job Streams and Output.
Microsoft.Automation/automationAccounts/jobs/suspend/action	Pause a job in progress.
Microsoft.Automation/automationAccounts/jobs/write	Create jobs.
Microsoft.Resources/subscriptions/resourceGroups/read	Read roles and role assignments.
Microsoft.Resources/deployments/*	Create and manage resource group deployments.
Microsoft.Insights/alertRules/*	Create and manage alert rules.
Microsoft.Support/*	Create and manage support tickets.

Automation Runbook Operator

An Automation Runbook Operator role is granted at the Runbook scope. An Automation Runbook Operator can view the runbook's name and properties. This role combined with the **Automation Job Operator** role enables the operator to also create and manage jobs for the runbook. The following table shows the permissions granted for the role:

ACTIONS	DESCRIPTION
Microsoft.Automation/automationAccounts/runbooks/read	List the runbooks.
Microsoft.Authorization/*/read	Read authorization.
Microsoft.Resources/subscriptions/resourceGroups/read	Read roles and role assignments.
Microsoft.Resources/deployments/*	Create and manage resource group deployments.
Microsoft.Insights/alertRules/*	Create and manage alert rules.
Microsoft.Support/*	Create and manage support tickets.

Log Analytics Contributor

A Log Analytics Contributor can read all monitoring data and edit monitoring settings. Editing monitoring settings includes adding the VM extension to VMs; reading storage account keys to be able to configure collection of logs from Azure Storage; creating and configuring Automation accounts; adding features; and configuring Azure diagnostics on all Azure resources. The following table shows the permissions granted for the role:

ACTIONS	DESCRIPTION
*/read	Read resources of all types, except secrets.
Microsoft.Automation/automationAccounts/*	Manage Automation accounts.
Microsoft.ClassicCompute/virtualMachines/extensions/*	Create and manage virtual machine extensions.
Microsoft.ClassicStorage/storageAccounts/listKeys/action	List classic storage account keys.
Microsoft.Compute/virtualMachines/extensions/*	Create and manage classic virtual machine extensions.
Microsoft.Insights/alertRules/*	Read/write/delete alert rules.
Microsoft.Insights/diagnosticSettings/*	Read/write/delete diagnostic settings.
Microsoft.OperationalInsights/*	Manage Azure Monitor logs.
Microsoft.OperationsManagement/*	Manage Azure Automation features in workspaces.
Microsoft.Resources/deployments/*	Create and manage resource group deployments.
Microsoft.Resources/subscriptions/resourcegroups/deployments/*	Create and manage resource group deployments.
Microsoft.Storage/storageAccounts/listKeys/action	List storage account keys.
Microsoft.Support/*	Create and manage support tickets.

Log Analytics Reader

A Log Analytics Reader can view and search all monitoring data as well as and view monitoring settings, including viewing the configuration of Azure diagnostics on all Azure resources. The following table shows the permissions granted or denied for the role:

ACTIONS	DESCRIPTION
*/read	Read resources of all types, except secrets.
Microsoft.OperationalInsights/workspaces/analytics/query/action	Manage queries in Azure Monitor logs.
Microsoft.OperationalInsights/workspaces/search/action	Search Azure Monitor log data.
Microsoft.Support/*	Create and manage support tickets.
Not Actions	
Microsoft.OperationalInsights/workspaces/sharedKeys/read	Not able to read the shared access keys.

Monitoring Contributor

A Monitoring Contributor can read all monitoring data and update monitoring settings. The following table shows the permissions granted for the role:

ACTIONS	DESCRIPTION
*/read	Read resources of all types, except secrets.
Microsoft.AlertsManagement/alerts/*	Manage Alerts.
Microsoft.AlertsManagement/alertsSummary/*	Manage the Alert dashboard.
Microsoft.Insights/AlertRules/*	Manage alert rules.
Microsoft.Insights/components/*	Manage Application Insights components.
Microsoft.Insights/DiagnosticSettings/*	Manage diagnostic settings.
Microsoft.Insights/eventtypes/*	List Activity Log events (management events) in a subscription. This permission is applicable to both programmatic and portal access to the Activity Log.
Microsoft.Insights/LogDefinitions/*	This permission is necessary for users who need access to Activity Logs via the portal. List log categories in Activity Log.
Microsoft.Insights/MetricDefinitions/*	Read metric definitions (list of available metric types for a resource).
Microsoft.Insights/Metrics/*	Read metrics for a resource.
Microsoft.Insights/Register/Action	Register the Microsoft.Insights provider.
Microsoft.Insights/webtests/*	Manage Application Insights web tests.
Microsoft.OperationalInsights/workspaces/intelligencepacks/*	Manage Azure Monitor logs solution packs.
Microsoft.OperationalInsights/workspaces/savedSearches/*	Manage Azure Monitor logs saved searches.
Microsoft.OperationalInsights/workspaces/search/action	Search Log Analytics workspaces.
Microsoft.OperationalInsights/workspaces/sharedKeys/action	List keys for a Log Analytics workspace.
Microsoft.OperationalInsights/workspaces/storageinsightconfigs/*	Manage Azure Monitor logs storage insight configurations.
Microsoft.Support/*	Create and manage support tickets.
Microsoft.WorkloadMonitor/workloads/*	Manage Workloads.

Monitoring Reader

A Monitoring Reader can read all monitoring data. The following table shows the permissions granted for the role:

ACTIONS	DESCRIPTION
*/read	Read resources of all types, except secrets.
Microsoft.OperationalInsights/workspaces/search/action	Search Log Analytics workspaces.
Microsoft.Support/*	Create and manage support tickets

User Access Administrator

A User Access Administrator can manage user access to Azure resources. The following table shows the permissions granted for the role:

ACTIONS	DESCRIPTION
*/read	Read all resources
Microsoft.Authorization/*	Manage authorization
Microsoft.Support/*	Create and manage support tickets

Feature setup permissions

The following sections describe the minimum required permissions needed for enabling the Update Management and Change Tracking and Inventory features.

Permissions for enabling Update Management and Change Tracking and Inventory from a VM

ACTION	PERMISSION	MINIMUM SCOPE
Write new deployment	Microsoft.Resources/deployments/*	Subscription
Write new resource group	Microsoft.Resources/subscriptions/resourceGroups/write	Subscription
Create new default Workspace	Microsoft.OperationalInsights工作spaces/write	Resource group
Create new Account	Microsoft.Automation/automationAccounts/write	Resource group
Link workspace and account	Microsoft.OperationalInsights/工作spaces/write Microsoft.Automation/automationAccounts/read	Workspace Automation account
Create MMA extension	Microsoft.Compute/virtualMachines/write	Virtual Machine
Create saved search	Microsoft.OperationalInsights/工作spaces/write	Workspace
Create scope config	Microsoft.OperationalInsights/工作spaces/write	Workspace

ACTION	PERMISSION	MINIMUM SCOPE
Onboarding state check - Read workspace	Microsoft.OperationalInsights/workspaces/read	Workspace
Onboarding state check - Read linked workspace property of account	Microsoft.Automation/automationAccounts/read	Automation account
Onboarding state check - Read solution	Microsoft.OperationalInsights/workspaces/intelligencepacks/read	Solution
Onboarding state check - Read VM	Microsoft.Compute/virtualMachines/read	Virtual Machine
Onboarding state check - Read account	Microsoft.Automation/automationAccounts/read	Automation account
Onboarding workspace check for VM ¹	Microsoft.OperationalInsights/workspaces/read	Subscription
Register the Log Analytics provider	Microsoft.Insights/register/action	Subscription

¹ This permission is needed to enable features through the VM portal experience.

Permissions for enabling Update Management and Change Tracking and Inventory from an Automation account

ACTION	PERMISSION	MINIMUM SCOPE
Create new deployment	Microsoft.Resources/deployments/*	Subscription
Create new resource group	Microsoft.Resources/subscriptions/resourceGroups/write	Subscription
AutomationOnboarding blade - Create new workspace	Microsoft.OperationalInsights/workspaces/write	Resource group
AutomationOnboarding blade - read linked workspace	Microsoft.Automation/automationAccounts/read	Automation account
AutomationOnboarding blade - read solution	Microsoft.OperationalInsights/workspaces/intelligencepacks/read	Solution
AutomationOnboarding blade - read workspace	Microsoft.OperationalInsights/workspaces/intelligencepacks/read	Workspace
Create link for workspace and Account	Microsoft.OperationalInsights/workspaces/write	Workspace
Write account for shoebox	Microsoft.Automation/automationAccounts/write	Account
Create/edit saved search	Microsoft.OperationalInsights/workspaces/write	Workspace

ACTION	PERMISSION	MINIMUM SCOPE
Create/edit scope config	Microsoft.OperationalInsights/workspaces/write	Workspace
Register the Log Analytics provider	Microsoft.Insights/register/action	Subscription
Step 2 - Enable Multiple VMs		
VMOnboarding blade - Create MMA extension	Microsoft.Compute/virtualMachines/write	Virtual Machine
Create / edit saved search	Microsoft.OperationalInsights/workspaces/write	Workspace
Create / edit scope config	Microsoft.OperationalInsights/workspaces/write	Workspace

Custom Azure Automation Contributor role

Microsoft intends to remove the Automation account rights from the Log Analytics Contributor role. Currently, the built-in [Log Analytics Contributor](#) role described above can escalate privileges to the subscription [Contributor](#) role. Since Automation account Run As accounts are initially configured with Contributor rights on the subscription, it can be used by an attacker to create new runbooks and execute code as a Contributor on the subscription.

As a result of this security risk, we recommend you don't use the Log Analytics Contributor role to execute Automation jobs. Instead, create the Azure Automation Contributor custom role and use it for actions related to the Automation account. Perform the following steps to create this custom role.

Create using the Azure portal

Perform the following steps to create the Azure Automation custom role in the Azure portal. If you would like to learn more, see [Azure custom roles](#).

1. Copy and paste the following JSON syntax into a file. Save the file on your local machine or in an Azure storage account. In the JSON file, replace the value for the `assignableScopes` property with the subscription GUID.

```
{
  "properties": {
    "roleName": "Automation Account Contributor (Custom)",
    "description": "Allows access to manage Azure Automation and its resources",
    "assignableScopes": [
      "/subscriptions/XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXX"
    ],
    "permissions": [
      {
        "actions": [
          "Microsoft.Authorization/*/read",
          "Microsoft.Insights/alertRules/*",
          "Microsoft.Insights/metrics/read",
          "Microsoft.Insights/diagnosticSettings/*",
          "Microsoft.Resources/deployments/*",
          "Microsoft.Resources/subscriptions/resourceGroups/read",
          "Microsoft.Automation/automationAccounts/*",
          "Microsoft.Support/*"
        ],
        "notActions": [],
        "dataActions": [],
        "notDataActions": []
      }
    ]
  }
}
```

2. Complete the remaining steps as outlined in [Create or update Azure custom roles using the Azure portal](#).

For [Step 3: Basics](#), note the following:

- In the **Custom role name** field, enter **Automation account Contributor (custom)** or a name matching your naming standards.
- For **Baseline permissions**, select **Start from JSON**. Then select the custom JSON file you saved earlier.

3. Complete the remaining steps, and then review and create the custom role. It can take a few minutes for your custom role to appear everywhere.

Create using PowerShell

Perform the following steps to create the Azure Automation custom role with PowerShell. If you would like to learn more, see [Azure custom roles](#).

1. Copy and paste the following JSON syntax into a file. Save the file on your local machine or in an Azure storage account. In the JSON file, replace the value for the **AssignableScopes** property with the subscription GUID.

```
{
    "Name": "Automation account Contributor (custom)",
    "Id": "",
    "IsCustom": true,
    "Description": "Allows access to manage Azure Automation and its resources",
    "Actions": [
        "Microsoft.Authorization/*/read",
        "Microsoft.Insights/alertRules/*",
        "Microsoft.Insights/metrics/read",
        "Microsoft.Insights/diagnosticSettings/*",
        "Microsoft.Resources/deployments/*",
        "Microsoft.Resources/subscriptions/resourceGroups/read",
        "Microsoft.Automation/automationAccounts/*",
        "Microsoft.Support/*"
    ],
    "NotActions": [],
    "AssignableScopes": [
        "/subscriptions/XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXX"
    ]
}
```

2. Complete the remaining steps as outlined in [Create or update Azure custom roles using Azure PowerShell](#). It can take a few minutes for your custom role to appear everywhere.

Update Management permissions

Update Management can be used to assess and schedule update deployments to machines in multiple subscriptions in the same Azure Active Directory (Azure AD) tenant, or across tenants using Azure Lighthouse. The following table lists the permissions needed to manage update deployments.

RESOURCE	ROLE	SCOPE
Automation account	Custom Azure Automation Contributor role	Automation account
Automation account	Virtual Machine Contributor	Resource Group for the account
Log Analytics workspace Log Analytics Contributor	Log Analytics workspace	
Log Analytics workspace	Log Analytics Reader	Subscription
Solution	Log Analytics Contributor	Solution
Virtual Machine	Virtual Machine Contributor	Virtual Machine
Actions on Virtual Machine		
View history of update schedule execution (Software Update Configuration Machine Runs)	Reader	Automation account
Actions on virtual machine		
Create update schedule (Software Update Configurations)	Microsoft.Compute/virtualMachines/write	For static VM list and resource groups

RESOURCE	ROLE	SCOPE
Create update schedule (Software Update Configurations)	Microsoft.OperationalInsights/workspaces/analytics/query/action	For workspace resource ID when using non-Azure dynamic list.

Configure Azure RBAC for your Automation account

The following section shows you how to configure Azure RBAC on your Automation account through the [Azure portal](#) and [PowerShell](#).

Configure Azure RBAC using the Azure portal

1. Log in to the [Azure portal](#) and open your Automation account from the Automation Accounts page.
2. Click on **Access control (IAM)** to open the Access control (IAM) page. You can use this page to add new users, groups, and applications to manage your Automation account and view existing roles that are configurable for the Automation account.
3. Click the **Role assignments** tab.

NAME	TYPE	ROLE	SCOPE
AzureJavaTe... App	App	Contributor	Subscription (Inherited)
ContosoAut... App	App	Contributor	Subscription (Inherited)
ContosoAut... App	App	Contributor	Subscription (Inherited)

Add a new user and assign a role

1. From the Access control (IAM) page, click **+ Add role assignment**. This action opens the Add role assignment page where you can add a user, group, or application, and assign a corresponding role.
2. Select a role from the list of available roles. You can choose any of the available built-in roles that an Automation account supports or any custom role you may have defined.
3. Type the name of the user that you want to give permissions to in the **Select** field. Choose the user from the list and click **Save**.

Add role assignment

Role Reader

Assign access to Azure AD user, group, or service principal

Select John Doe

Selected members: John Doe

Save **Discard**

Now you should see the user added to the Users page, with the selected role assigned.

Add role assignment **Edit columns** **Refresh** **Remove**

Role assignments

Manage access to Azure resources for users, groups, and service principals at this scope by creating role assignments. [Learn More](#)

Name	Type	Role
Search by name or email	All	Reader
Scope	Group by	
All scopes	Role	

Showing a filtered set of results. Total number of role assignments: 15

1 items (1 Users)

<input type="checkbox"/> NAME	TYPE	ROLE	SCOPE
READER			
JD John Doe John.Doe@example.com	User	Reader	This resource

You can also assign a role to the user from the Roles page.

- Click **Roles** from the Access control (IAM) page to open the Roles page. You can view the name of the role and the number of users and groups assigned to that role.

Roles

TestAzureAuto

NAME	USERS	GROUPS
Owner ⓘ	0	1
Contributor ⓘ	12	0
Reader ⓘ	0	0
Automation Operator ⓘ	0	0
Azure Service Deploy Release Management Contributor ⓘ	0	0
Log Analytics Contributor ⓘ	0	0
Log Analytics Reader ⓘ	0	0
masterreader ⓘ	0	0
Monitoring Contributor ⓘ	0	0
Monitoring Reader ⓘ	0	0
Resource Policy Contributor (Preview) ⓘ	0	0
User Access Administrator ⓘ	0	0

NOTE

You can only set role-based access control at the Automation account scope and not at any resource below the Automation account.

Remove a user

You can remove the access permission for a user who isn't managing the Automation account, or who no longer works for the organization. Following are the steps to remove a user:

1. From the Access control (IAM) page, select the user to remove and click **Remove**.
2. Click the **Remove** button in the assignment details pane.
3. Click **Yes** to confirm removal.

Remove role assignments
Are you sure you want to remove the selected assigned role assignments?

Yes **No**

Configure Azure RBAC using PowerShell

You can also configure role-based access to an Automation account using the following [Azure PowerShell cmdlets](#):

`Get-AzRoleDefinition` lists all Azure roles that are available in Azure Active Directory. You can use this cmdlet with the `Name` parameter to list all the actions that a specific role can perform.

```
Get-AzRoleDefinition -Name 'Automation Operator'
```

The following is the example output:

```
Name      : Automation Operator
Id       : d3881f73-407a-4167-8283-e981cbba0404
IsCustom   : False
Description : Automation Operators are able to start, stop, suspend, and resume jobs
Actions    : {Microsoft.Authorization/*/read, Microsoft.Automation/automationAccounts/jobs/read,
              Microsoft.Automation/automationAccounts/jobs/resume/action,
              Microsoft.Automation/automationAccounts/jobs/stop/action...}
NotActions  : {}
AssignableScopes : {}
```

[Get-AzRoleAssignment](#) lists Azure role assignments at the specified scope. Without any parameters, this cmdlet returns all the role assignments made under the subscription. Use the `ExpandPrincipalGroups` parameter to list access assignments for the specified user, as well as the groups that the user belongs to.

Example: Use the following cmdlet to list all the users and their roles within an Automation account.

```
Get-AzRoleAssignment -Scope '/subscriptions/<SubscriptionID>/resourcegroups/<Resource Group Name>/Providers/Microsoft.Automation/automationAccounts/<Automation account name>'
```

The following is the example output:

```
RoleAssignmentId  : /subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/myResourceGroup/providers/Microsoft.Automation/automationAccounts/myAutomationAc
count/provid
ers/Microsoft.Authorization/roleAssignments/cc594d39-ac10-46c4-9505-f182a355c41f
Scope      : /subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/myResourceGroup/providers/Microsoft.Automation/automationAccounts/myAutomationAc
count
DisplayName     : admin@contoso.com
SignInName      : admin@contoso.com
RoleDefinitionName : Automation Operator
RoleDefinitionId  : d3881f73-407a-4167-8283-e981cbba0404
ObjectId        : 15f26a47-812d-489a-8197-3d4853558347
ObjectType      : User
```

Use [New-AzRoleAssignment](#) to assign access to users, groups, and applications to a particular scope.

Example: Use the following command to assign the "Automation Operator" role for a user in the Automation account scope.

```
New-AzRoleAssignment -SignInName <sign-in Id of a user you wish to grant access> -RoleDefinitionName
'Automation operator' -Scope '/subscriptions/<SubscriptionID>/resourcegroups/<Resource Group Name>/Providers/Microsoft.Automation/automationAccounts/<Automation account name>'
```

The following is the example output:

```

RoleAssignmentId    : /subscriptions/00000000-0000-0000-0000-
000000000000/resourcegroups/myResourceGroup/Providers/Microsoft.Automation/automationAccounts/myAutomationAc
count/provid
Scope              : /subscriptions/00000000-0000-0000-0000-
000000000000/resourcegroups/myResourceGroup/Providers/Microsoft.Automation/automationAccounts/myAutomationAc
count
DisplayName        : admin@contoso.com
SignInName         : admin@contoso.com
RoleDefinitionName : Automation Operator
RoleDefinitionId   : d3881f73-407a-4167-8283-e981cbba0404
ObjectId           : f5ecbe87-1181-43d2-88d5-a8f5e9d8014e
ObjectType         : User

```

Use [Remove-AzRoleAssignment](#) to remove access of a specified user, group, or application from a particular scope.

Example: Use the following command to remove the user from the Automation Operator role in the Automation account scope.

```

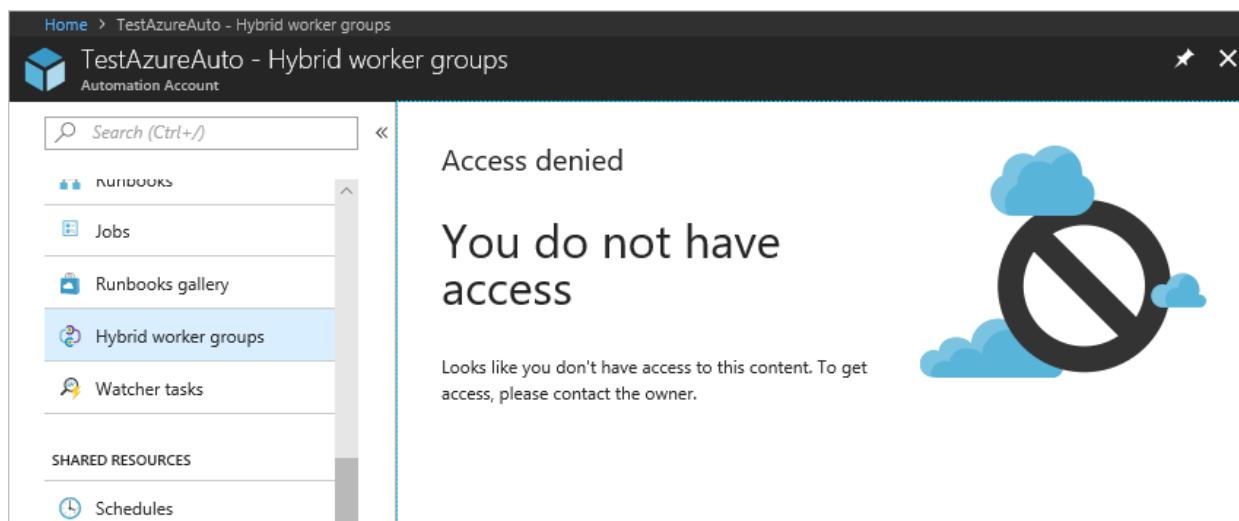
Remove-AzRoleAssignment -SignInName <sign-in ID of a user you wish to remove> -RoleDefinitionName
'Automation Operator' -Scope '/subscriptions/<SubscriptionID>/resourcegroups/<Resource Group
Name>/Providers/Microsoft.Automation/automationAccounts/<Automation account name>'

```

In the preceding example, replace `sign-in ID of a user you wish to remove`, `SubscriptionID`, `Resource Group Name`, and `Automation account name` with your account details. Choose `yes` when prompted to confirm before continuing to remove user role assignments.

User experience for Automation Operator role - Automation account

When a user assigned to the Automation Operator role on the Automation account scope views the Automation account to which he/she is assigned, the user can only view the list of runbooks, runbook jobs, and schedules created in the Automation account. This user can't view the definitions of these items. The user can start, stop, suspend, resume, or schedule the runbook job. However, the user doesn't have access to other Automation resources, such as configurations, Hybrid Runbook Worker groups, or DSC nodes.



Configure Azure RBAC for runbooks

Azure Automation allows you to assign Azure roles to specific runbooks. To do this, run the following script to add a user to a specific runbook. An Automation Account Administrator or a Tenant Administrator can run this script.

```

$rgName = "<Resource Group Name>" # Resource Group name for the Automation account
$automationAccountName = "<Automation account name>" # Name of the Automation account
$rbName = "<Name of Runbook>" # Name of the runbook
$userID = "<User ObjectId>" # Azure Active Directory (AAD) user's ObjectId from the directory

# Gets the Automation account resource
$aa = Get-AzResource -ResourceGroupName $rgName -ResourceType "Microsoft.Automation/automationAccounts" -ResourceName $automationAccountName

# Get the Runbook resource
$rb = Get-AzResource -ResourceGroupName $rgName -ResourceType "Microsoft.Automation/automationAccounts/runbooks" -ResourceName "$rbName"

# The Automation Job Operator role only needs to be run once per user.
New-AzRoleAssignment -ObjectId $userID -RoleDefinitionName "Automation Job Operator" -Scope $aa.ResourceId

# Adds the user to the Automation Runbook Operator role to the Runbook scope
New-AzRoleAssignment -ObjectId $userID -RoleDefinitionName "Automation Runbook Operator" -Scope $rb.ResourceId

```

Once the script has run, have the user log in to the Azure portal and select **All Resources**. In the list, the user can see the runbook for which he/she has been added as an Automation Runbook Operator.

The screenshot shows the Azure portal's 'All resources' blade. At the top, there are navigation links for 'Home > All resources' and a subscription dropdown set to 'Contoso (contoso.onmicrosoft.com)'. Below the header are buttons for 'Add', 'Edit columns', 'Refresh', 'Assign Tags', and 'Delete'. A search bar is followed by filters for 'Filter by name...', 'All resource groups', 'All types', 'All locations', and 'No grouping'. A summary shows '1 items' found. The main table lists the single resource: 'Hello-World (TestAzureAutomationAc...' (Runbook), under 'Contoso Resource...', in East US, and associated with 'Contoso Subscription'. The table has columns for NAME, TYPE, RESOURCE GROUP, LOCATION, and SUBSCRIPTION.

User experience for Automation operator role - Runbook

When a user assigned to the Automation Operator role on the Runbook scope views an assigned runbook, the user can only start the runbook and view the runbook jobs.

The screenshot shows the Azure portal's 'Hello-World' runbook details page. The top navigation bar includes 'Home > Hello-World' and a 'Runbook' icon. The main content area has tabs for 'Start', 'View', 'Edit', 'Schedule', 'Webhook', 'Delete', and 'More'. On the left, a sidebar lists 'Overview', 'Activity log', 'Tags', 'Diagnose and solve problems', 'RESOURCES' (Jobs, Schedules, Webhooks), and 'RUNBOOK SETTINGS' (Properties). The main panel displays a message 'No access' and a 'Details' section with a 'Jobs' icon. The overall interface is clean and modern, typical of the Azure portal's design.

Next steps

- To find out more about Azure RBAC using PowerShell, see [Add or remove Azure role assignments using Azure PowerShell](#).
- For details of the types of runbooks, see [Azure Automation runbook types](#).
- To start a runbook, see [Start a runbook in Azure Automation](#).

Move your Azure Automation account to another subscription

4/22/2021 • 4 minutes to read • [Edit Online](#)

Azure Automation allows you to move some resources to a new resource group or subscription. You can move resources through the Azure portal, PowerShell, the Azure CLI, or the REST API. To learn more about the process, see [Move resources to a new resource group or subscription](#).

The Automation account is one of the resources that you can move. In this article, you'll learn to move Automation accounts to another resource or subscription. The high-level steps for moving your Automation account are:

1. Disable your features.
2. Unlink your workspace.
3. Move the Automation account.
4. Delete and re-create the Run As accounts.
5. Re-enable your features.

Remove features

To unlink your workspace from your Automation account, you must remove the feature resources in your workspace:

- Change Tracking and Inventory
- Update Management
- Start/Stop VMs during off-hours

1. In the Azure portal, locate your resource group.
2. Find each feature, and select **Delete** on the **Delete Resources** page.

The screenshot shows the Azure portal interface for managing a resource group named 'examplegroup'. On the left, there's a navigation menu with sections like Overview, Activity log, Access control (IAM), Tags, Events, Settings, Deployments, Policies, Properties, Locks, Automation script, Monitoring (with Insights and Alerts), and Metrics. The main area displays subscription information (Subscription ID: 00000000-0000-0000-0000-000000000000) and tags. A 'Delete Resources' dialog is open, asking if you want to delete all selected resources. It includes a warning message about the irreversibility of the action, a 'Type 'Yes'' confirmation input field, and a list of selected resources under 'Selected resources'.

If you prefer, you can delete the resources by using the [Remove-AzResource](#) cmdlet:

```
$workspaceName = <myWorkspaceName>
$resourceGroupName = <myResourceGroup>
Remove-AzResource -ResourceType 'Microsoft.OperationsManagement/solutions' -ResourceName "ChangeTracking($workspaceName)" -ResourceGroupName $resourceGroupName
Remove-AzResource -ResourceType 'Microsoft.OperationsManagement/solutions' -ResourceName "Updates($workspaceName)" -ResourceGroupName $resourceGroupName
Remove-AzResource -ResourceType 'Microsoft.OperationsManagement/solutions' -ResourceName "Start-Stop-VM($workspaceName)" -ResourceGroupName $resourceGroupName
```

Remove alert rules for Start/Stop VMs during off-hours

For Start/Stop VMs during off-hours, you also need to remove the alert rules created by the feature.

1. In the Azure portal, go to your resource group and select **Monitoring > Alerts > Manage alert rules**.

examplegroup - Alerts

Resource group

New alert rule **Manage alert rules** **Manage action groups** **View classic alerts** **Refresh**

Don't see a subscription? [Open Directory + Subscription settings](#)

* Subscription: Microsoft Azure | Resource group: examplegroup | Resource: Type to start filtering ... | Time range: Past 24 hours

Microsoft Azure > examplegroup

All is good! You have no alerts.

② **Manage alert rules (3)**

The classic alerts can be accessed from [here](#).

2. On the Rules page, you should see a list of the alerts configured in that resource group. The feature creates these rules:
 - AutoStop_VM_Child
 - ScheduledStartStop_Parent
 - SequencedStartStop_Parent
3. Select the rules one at a time, and select **Delete** to remove them.

Rules

Rules management

New alert rule **Edit columns** **Manage action groups** **View classic alerts** **Refresh** **Enable** **Disable** **Delete**

Delete

Are you sure you want to delete ?

Yes **No**

Click on "View classic alerts" to view rules configured in Alerts (classic).

Displaying 1 - 3 rules out of total 3 rules

Search alert rules based on rule name and condition...

NAME	CONDITION	STATUS	TARGET RESOURCE	TARGET RESOURCE ...	SIGNAL TYPE
AutoStop_VM_Child	AzureDiagnostics where (RunbookName_s == "AutoStop_V...)	Disabled	contosofinanceblogs	Log Analytics worksp...	Log Search
ScheduledStartStop_Parent	AzureDiagnostics where (RunbookName_s == "ScheduledS...)	Disabled	contosofinanceblogs	Log Analytics worksp...	Log Search
SequencedStartStop_Parent	AzureDiagnostics where (RunbookName_s == "SequencedS...)	Disabled	contosofinanceblogs	Log Analytics worksp...	Log Search

NOTE

If you don't see any alert rules on the Rules page, change the **Status** field to **Disabled** to show disabled alerts.

4. When you remove the alert rules, you must remove the action group created for Start/Stop VMs during off-hours notifications. In the Azure portal, select **Monitor > Alerts > Manage action groups**.
5. Select **StartStop_VM_Notification**.
6. On the action group page, select **Delete**.

StartStop_VM_Notification

Save Discard Refresh Delete

* Short name: StStAlert

Action group name: StartStop_VM_Notification

Resource group: examplegroup

Subscription: Microsoft Azure

Actions

ACTION NAME	ACTION TYPE	STATUS	DETAILS	ACTIONS
EmailInt3m3rhm56gq4	Email/SMS/Push/V...	Subscribed	Edit details	X
Unique name for the action	▼			

[Privacy Statement](#)

[Pricing](#)

If you prefer, you can delete your action group by using the [Remove-AzActionGroup](#) cmdlet:

```
Remove-AzActionGroup -ResourceGroupName <myResourceGroup> -Name StartStop_VM_Notification
```

Unlink your workspace

Now you can unlink your workspace:

1. In the Azure portal, select **Automation account** > **Related Resources** > **Linked workspace**.
2. Select **Unlink workspace** to unlink the workspace from your Automation account.

ContosoFinance - Linked workspace

Automation Account

Search (Ctrl+ /)

Go to workspace Unlink workspace

Related Resources

- Linked workspace
- Event grid
- Start/Stop VM

Account Settings

- Properties
- Keys
- Pricing
- Source control
- Run as accounts

Settings

- Locks
- Automation script

Linked workspace

This Automation account is linked to the following Log Analytics workspace: contosofinancelogs

Unlink workspace

To unlink this Automation account and the Log Analytics workspace you must first remove solutions that have a dependency on Automation from the workspace.
These are the following:

- Update Management
- Change Tracking
- Start/Stop VMs during off-hours

After you remove these solutions you can click **Unlink workspace** above to complete the unlinking.

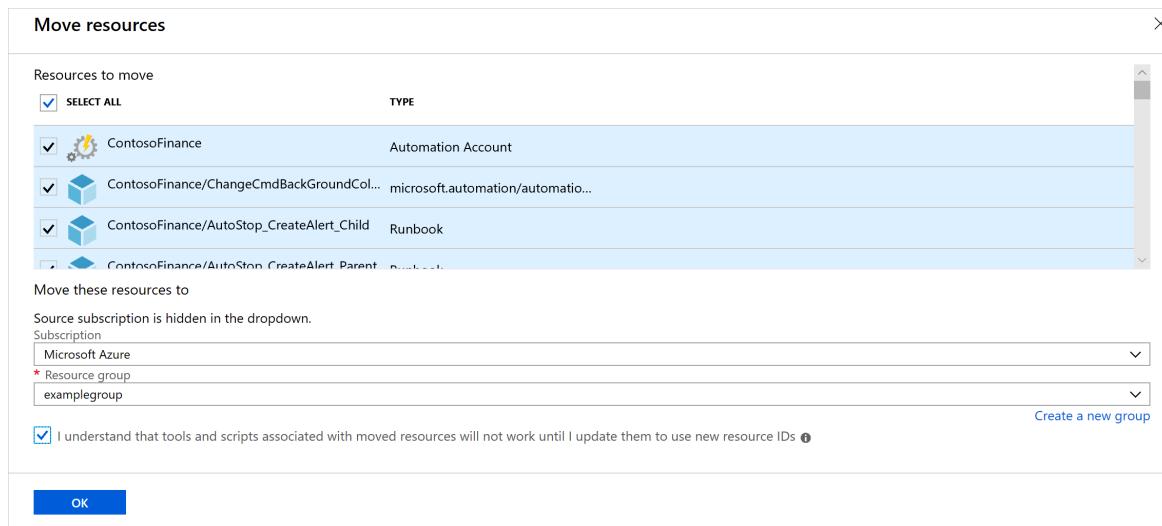
If you use the Update Management solution you optionally may want to remove some items that are no longer needed after you remove the solution.

- Update schedules (will have names that match the names of the update deployments you created)
[View schedules](#)
- Hybrid worker groups created for the solution (will have names like machine1.contoso.com_9ceb8108-26c9-4051-b6b3-227600d715c8).
[Learn how to remove a hybrid worker group.](#)

Move your Automation account

You can now move your Automation account and its runbooks.

1. In the Azure portal, browse to the resource group of your Automation account. Select **Move > Move to another subscription**.



2. Select the resources in your resource group that you want to move. Ensure that you include your Automation account, runbooks, and Log Analytics workspace resources.

Re-create Run As accounts

[Run As accounts](#) create a service principal in Azure Active Directory to authenticate with Azure resources. When you change subscriptions, the Automation account no longer uses the existing Run As account. To re-create the Run As accounts:

1. Go to your Automation account in the new subscription, and select **Run as accounts** under **Account Settings**. You'll see that the Run As accounts show as incomplete now.

2. Delete the Run As accounts, one at a time, by selecting **Delete** on the **Properties** page.

NOTE

If you don't have permissions to create or view the Run As accounts, you see the following message:

You do not have permissions to create an Azure Run As account (service principal) and grant the Contributor role to the service principal.

For more information, see [Permissions required to configure Run As accounts](#).

3. After you've deleted the Run As accounts, select **Create** under **Azure Run As account**.
4. On the Add Azure Run As account page, select **Create** to create the Run As account and service principal.
5. Repeat the steps above with the Azure Classic Run As account.

Enable features

After you re-create the Run As accounts, you must re-enable the features that you removed before the move:

1. To turn on Change Tracking and Inventory, select **Change Tracking and Inventory** in your Automation account. Choose the Log Analytics workspace that you moved over, and select **Enable**.

2. Repeat step 1 for Update Management.

The screenshot shows the 'ContosoFinance - Update management' configuration page. On the left, there's a sidebar with 'Automation Account' and sections for 'Change tracking', 'State configuration (DSC)', 'Update management' (which is selected and highlighted with a red box), 'Process Automation' (with 'Runbooks', 'Jobs', 'Runbooks gallery', 'Hybrid worker groups', and 'Watcher tasks'), and 'Shared Resources' (with 'Schedules'). The main area has a heading 'Enable consistent control and compliance of your VMs with Update Management.' Below it, it says 'This service is included with Azure virtual machines. You only pay for logs stored in Log Analytics.' and 'This service requires a Log Analytics workspace and this Automation account.' It includes fields for 'Location' (set to 'East US'), 'Log Analytics workspace subscription' (set to 'Contoso'), 'Log Analytics workspace' (highlighted with a red box, showing 'ContosoFinanceLogs' selected), and 'Automation account' (set to 'ContosoFinance'). At the bottom is a large blue 'Enable' button.

3. Machines that are enabled with your features are visible when you've connected the existing Log Analytics workspace. To turn on the Start/Stop VMs during off-hours feature, you must re-enable it. Under **Related Resources**, select **Start/Stop VMs** > **Learn more about and enable the solution** > **Create** to start the deployment.

4. On the Add Solution page, choose your Log Analytics workspace and Automation account.

The screenshot shows the 'Add Solution' page. It has a header 'Add Solution' with a close button. Below it, there are two main sections: 'Workspace' (containing 'ContosoFinanceLogs') and 'Automation account' (containing 'ContosoFinance'). A 'Configuration' section is expanded, showing 'Configure parameters' (with 'Parameters' listed). At the bottom, there's a note: 'Note: It might take up to several minutes to create the solution. Please check portal notifications for the progress.' Finally, there's a large blue 'Create' button at the bottom.

5. Configure the feature as described in [Start/Stop VMs during off-hours overview](#).

Verify the move

When the move is complete, verify that the capabilities listed below are enabled.

CAPABILITY	TESTS	TROUBLESHOOTING
Runbooks	A runbook can successfully run and connect to Azure resources.	Troubleshoot runbooks
Source control	You can run a manual sync on your source control repository.	Source control integration
Change tracking and inventory	Verify that you see current inventory data from your machines.	Troubleshoot change tracking
Update management	Verify that you see your machines and that they're healthy. Run a test software update deployment.	Troubleshoot update management
Shared resources	Verify that you see all your shared resources, such as credentials and variables .	

Next steps

To learn about moving resources in Azure, see [Move resources in Azure](#).

How to delete your Azure Automation account

6/8/2021 • 5 minutes to read • [Edit Online](#)

After you enable an Azure Automation account to help automate IT or business process, or enable its other features to support operations management of your Azure and non-Azure machines such as Update Management, you may decide to stop using the Automation account. If you have enabled features that depend on integration with an Azure Monitor Log Analytics workspace, there are more steps required to complete this action.

Removing your Automation account can be done using one of the following methods based on the supported deployment models:

- Delete the resource group containing the Automation account.
- Delete the resource group containing the Automation account and linked Azure Monitor Log Analytics workspace, if:
 - The account and workspace is dedicated to supporting Update Management, Change Tracking, and Inventory, and/or Start/Stop VMs during off-hours.
 - The account is dedicated to process automation and integrated with a workspace to send runbook job status and job streams.
- Unlink the Log Analytics workspace from the Automation account and delete the Automation account.
- Delete the feature from your linked workspace, unlink the account from the workspace, and then delete the Automation account.

This article tells you how to completely remove your Automation account through the Azure portal, using Azure PowerShell, the Azure CLI, or the REST API.

Prerequisite

Verify there aren't any [Resource Manager locks](#) applied at the subscription, resource group, or resource, which prevents accidental deletion or modification of critical resources. If you've deployed the Start/Stop VMs during off-hours solution, it sets the lock level to **CanNotDelete** against several dependent resources in the Automation account (specifically its runbooks and variables). Remove any locks before deleting the Automation account.

NOTE

If you receive an error message similar to:

The link cannot be updated or deleted because it is linked to Update Management and/or ChangeTracking Solutions

, then your Automation account is linked to a Log Analytics workspace with either the Update Management and/or Change Tracking and Inventory features enabled. For more information, see [Delete a shared capability Automation account](#), below.

Delete the dedicated resource group

To delete your Automation account, and also the Log Analytics workspace if linked to the account, created in the same resource group dedicated to the account, follow the steps outlined in the [Azure Resource Manager resource group and resource deletion](#) article.

Delete a standalone Automation account

If your Automation account isn't linked to a Log Analytics workspace, perform the following steps to delete it.

- [Azure portal](#)
- [PowerShell](#)

1. Sign in to Azure at <https://portal.azure.com>.
2. In the Azure portal, navigate to **Automation Accounts**.
3. Open your Automation account and select **Delete** from the menu.

While the information is verified and the account is deleted, you can track the progress under **Notifications**, chosen from the menu.

Delete a standalone Automation account linked to workspace

If your Automation account is linked to a Log Analytics workspace to collect job streams and job logs, perform the following steps to delete the account.

There are two options for unlinking the Log Analytics workspace from your Automation account. You can perform this process from the Automation account or from the linked workspace.

To unlink from your Automation account, perform the following steps.

1. In the Azure portal, navigate to **Automation Accounts**.
2. Open your Automation account and select **Linked workspace** under **Related Resources** on the left.
3. On the **Unlink workspace** page, select **Unlink workspace**, and respond to prompts.

The screenshot shows the Azure portal interface for managing an Automation account. The main title is "MAIC-AA-Pri | Linked workspace". The left sidebar lists "Related Resources" such as "Linked workspace", "Event grid", and "Start/Stop VM". The "Account Settings" section includes "Properties", "Keys", and "Pricing". The main content area shows a "Linked workspace" section with a link to "maic-la". Below this, there's a "Unlink workspace" button and a note about removing linked resources like Update Management, Change Tracking, and Start/Stop VMs during off-hours. A detailed note explains the process of unlinking after removing these resources.

While it attempts to unlink the Log Analytics workspace, you can track the progress under **Notifications** from the menu.

To unlink from the workspace, perform the following steps.

1. In the Azure portal, navigate to **Log Analytics workspaces**.

- From the workspace, select **Automation Account** under **Related Resources**.
- On the Automation Account page, select **Unlink account**, and respond to prompts.

While it attempts to unlink the Automation account, you can track the progress under **Notifications** from the menu.

After the Automation account is successfully unlinked from the workspace, perform the steps in the [standalone Automation account](#) section to delete the account.

Delete a shared capability Automation account

To delete your Automation account linked to a Log Analytics workspace in support of Update Management, Change Tracking and Inventory, and/or Start/Stop VMs during off-hours, perform the following steps.

Step 1. Delete the solution from the linked workspace

- [Azure portal](#)
- [PowerShell](#)

- Sign in to Azure at <https://portal.azure.com>.
- Navigate to your Automation account, and select **Linked workspace** under **Related resources**.
- Select **Go to workspace**.
- Select **Solutions** under **General**.
- On the Solutions page, select one of the following based on the feature(s) deployed in the account:
 - For Start/Stop VMs during off-hours, select **Start-Stop-VM[workspace name]**.
 - For Update Management, select **Updates(workspace name)**.
 - For Change Tracking and Inventory, select **ChangeTracking(workspace name)**.
- On the **Solution** page, select **Delete** from the menu. If more than one of the above listed features are deployed to the Automation account and linked workspace, you need to select and delete each one before proceeding.
- While the information is verified and the feature is deleted, you can track the progress under **Notifications**, chosen from the menu. You're returned to the Solutions page after the removal process.

Step 2. Unlink workspace from Automation account

There are two options for unlinking the Log Analytics workspace from your Automation account. You can perform this process from the Automation account or from the linked workspace.

To unlink from your Automation account, perform the following steps.

- In the Azure portal, navigate to **Automation Accounts**.
- Open your Automation account and select **Linked workspace** under **Related Resources** on the left.
- On the **Unlink workspace** page, select **Unlink workspace**, and respond to prompts.

Home > Automation Accounts >

MAIC-AA-Pri | Linked workspace

Automation Account

Search (Ctrl+ /) Go to workspace Unlink workspace

Credentials Connections Certificates Variables

Related Resources

- Linked workspace
- Event grid
- Start/Stop VM

Account Settings

- Properties
- Keys
- Pricing

Linked workspace

This Automation account is linked to the following Log Analytics workspace: [maic-la](#)

Unlink workspace

To unlink this Automation account and the Log Analytics workspace you must first remove These are the following:

- Update Management
- Change Tracking
- Start/Stop VMs during off-hours

After you remove these solutions you can click **Unlink workspace** above to complete the process.

If you use the Update Management solution you optionally may want to remove some items:

- Update schedules (will have names that match the names of the update deployment groups)
[View schedules](#)
- Hybrid worker groups created for the solution (will have names like machine1.contoso.com)

[Learn how to remove a hybrid worker group.](#)

While it attempts to unlink the Log Analytics workspace, you can track the progress under **Notifications** from the menu.

To unlink from the workspace, perform the following steps.

1. In the Azure portal, navigate to **Log Analytics workspaces**.
2. From the workspace, select **Automation Account** under **Related Resources**.
3. On the Automation Account page, select **Unlink account**, and respond to prompts.

While it attempts to unlink the Automation account, you can track the progress under **Notifications** from the menu.

Step 3. Delete Automation account

After the Automation account is successfully unlinked from the workspace, perform the steps in the [standalone Automation account](#) section to delete the account.

Next steps

To create an Automation account from the Azure portal, see [Create a standalone Azure Automation account](#). If you prefer to create your account using a template, see [Create an Automation account using an Azure Resource Manager template](#).

Migrate from Orchestrator to Azure Automation (Beta)

4/22/2021 • 9 minutes to read • [Edit Online](#)

Runbooks in [System Center 2012 - Orchestrator](#) are based on activities from integration packs that are written specifically for Orchestrator, while runbooks in Azure Automation are based on Windows PowerShell. [Graphical runbooks](#) in Azure Automation have a similar appearance to Orchestrator runbooks, with their activities representing PowerShell cmdlets, child runbooks, and assets. In addition to converting runbooks themselves, you must convert the integration packs with the activities that the runbooks use to integration modules with Windows PowerShell cmdlets.

[Service Management Automation](#) (SMA) stores and runs runbooks in your local datacenter like Orchestrator, and it uses the same integration modules as Azure Automation. The Runbook Converter converts Orchestrator runbooks to graphical runbooks, which are not supported in SMA. You can still install the Standard Activities Module and System Center Orchestrator Integration Modules into SMA, but you must manually [rewrite your runbooks](#).

Download the Orchestrator migration toolkit

The first step in migration is to download the [System Center Orchestrator Migration Toolkit](#). This toolkit includes tools to assist you in converting runbooks from Orchestrator to Azure Automation.

Import the Standard Activities module

Import the [Standard Activities Module](#) into Azure Automation. This includes converted versions of standard Orchestrator activities that converted graphical runbooks can use.

Import Orchestrator integration modules

Microsoft provides [integration packs](#) for building runbooks to automate System Center components and other products. Some of these integration packs are currently based on OIT but cannot currently be converted to integration modules because of known issues. Import [System Center Orchestrator Integration Modules](#) into Azure Automation for the integration packs used by your runbooks that access System Center. This package includes converted versions of the integration packs that can be imported into Azure Automation and Service Management Automation.

Convert integration packs

Use the [Integration Pack Converter](#) to convert any integration packs created using the [Orchestrator Integration Toolkit \(OIT\)](#) to PowerShell-based integration modules that can be imported into Azure Automation or Service Management Automation. When you run the Integration Pack Converter, you are presented with a wizard that allows you to select an integration pack (.oip) file. The wizard then lists the activities included in that integration pack and allows you to select which activities to migrate. When you complete the wizard, it creates an integration module that includes a corresponding cmdlet for each of the activities in the original integration pack.

NOTE

You can't use the Integration Pack Converter to convert integration packs that were not created with OIT. There are also some integration packs provided by Microsoft that can't currently be converted with this tool. Converted versions of these integration packs are provided for download so that they can be installed in Azure Automation or Service Management Automation.

Parameters

Any properties of an activity in the integration pack are converted to parameters of the corresponding cmdlet in the integration module. Windows PowerShell cmdlets have a set of [common parameters](#) that can be used with all cmdlets. For example, the `-Verbose` parameter causes a cmdlet to output detailed information about its operation. No cmdlet may have a parameter with the same name as a common parameter. If an activity does have a property with the same name as a common parameter, the wizard prompts you to provide another name for the parameter.

Monitor activities

Monitor runbooks in Orchestrator start with a [monitor activity](#) and run continuously waiting to be invoked by a particular event. Azure Automation does not support monitor runbooks, so any monitor activities in the integration pack is not converted. Instead, a placeholder cmdlet is created in the integration module for the monitor activity. This cmdlet has no functionality, but it allows any converted runbook that uses it to be installed. This runbook is not able to run in Azure Automation, but it can be installed so that you can modify it.

Orchestrator includes a set of [standard activities](#) that are not included in an integration pack but are used by many runbooks. The Standard Activities module is an integration module that includes a cmdlet equivalent for each of these activities. You must install this integration module in Azure Automation before importing any converted runbooks that use a standard activity.

In addition to supporting converted runbooks, the cmdlets in the standard activities module can be used by someone familiar with Orchestrator to build new runbooks in Azure Automation. While the functionality of all of the standard activities can be performed with cmdlets, they may operate differently. The cmdlets in the converted standard activities module work in the same way as their corresponding activities and use the same parameters. This can help you in transitioning to Azure Automation runbooks.

Convert Orchestrator runbooks

The Orchestrator Runbook Converter converts Orchestrator runbooks into [graphical runbooks](#) that can be imported into Azure Automation. The Runbook Converter is implemented as a PowerShell module with the cmdlet `ConvertFrom-SCORunbook` that makes the conversion. When you install the converter, it creates a shortcut to a PowerShell session that loads the cmdlet.

Here are the basic steps to convert a runbook and import it into Azure Automation. Details of using the cmdlet are provided later in this section.

1. Export one or more runbooks from Orchestrator.
2. Obtain integration modules for all activities in the runbook.
3. Convert the Orchestrator runbooks in the exported file.
4. Review information in logs to validate the conversion and to determine any required manual tasks.
5. Import converted runbooks into Azure Automation.
6. Create any required assets in Azure Automation.
7. Edit the runbook in Azure Automation to modify any required activities.

The syntax for `ConvertFrom-SCORunbook` is:

```
ConvertFrom-SCORunbook -RunbookPath <string> -Module <string[]> -OutputFolder <string>
```

- RunbookPath - Path to the export file containing the runbooks to convert.
- Module - Comma delimited list of integration modules containing activities in the runbooks.
- OutputFolder - Path to the folder to create converted graphical runbooks.

The following example command converts the runbooks in an export file called **MyRunbooks.ois_export**. These runbooks use the Active Directory and Data Protection Manager integration packs.

```
ConvertFrom-SCORunbook -RunbookPath "c:\runbooks\MyRunbooks.ois_export" -Module  
c:\ip\SystemCenter_IntegrationModule_ActiveDirectory.zip,c:\ip\SystemCenter_IntegrationModule_DPM.zip -  
OutputFolder "c:\runbooks"
```

Use Runbook Converter log files

The Runbook Converter creates the following log files in the same location as the converted runbook. If the files already exist, they are overwritten with information from the last conversion.

FILE	CONTENTS
Runbook Converter - Progress.log	Detailed steps of the conversion including information for each activity successfully converted and warning for each activity not converted.
Runbook Converter - Summary.log	Summary of the last conversion including any warnings and follow up tasks that you need to perform such as creating a variable required for the converted runbook.

Export runbooks from Orchestrator

The Runbook Converter works with an export file from Orchestrator that contains one or more runbooks. It creates a corresponding Azure Automation runbook for each Orchestrator runbook in the export file.

To export a runbook from Orchestrator, right-click the name of the runbook in Runbook Designer and select **Export**. To export all runbooks in a folder, right-click the name of the folder and select **Export**.

Convert runbook activities

The Runbook Converter converts each activity in the Orchestrator runbook to a corresponding activity in Azure Automation. For those activities that can't be converted, a placeholder activity is created in the runbook with warning text. After you import the converted runbook into Azure Automation, you must replace any of these activities with valid activities that perform the required functionality.

Any Orchestrator activities in the Standard Activities Module are converted. There are some standard Orchestrator activities that are not in this module though and are not converted. For example,

`Send Platform Event` has no Azure Automation equivalent since the event is specific to Orchestrator.

Monitor activities are not converted since there is no equivalent to them in Azure Automation. The exceptions are monitor activities in converted integration packs that are converted to the placeholder activity.

Any activity from a converted integration pack is converted if you provide the path to the integration module with the `modules` parameter. For System Center Integration Packs, you can use the System Center Orchestrator Integration Modules.

Manage Orchestrator resources

The Runbook Converter only converts runbooks, not other Orchestrator resources such as counters, variables, or connections. Counters are not supported in Azure Automation. Variables and connections are supported, but

you must create them manually. The log files inform you if the runbook requires such resources and specify corresponding resources that you need to create in Azure Automation for the converted runbook to operate properly.

For example, a runbook may use a variable to populate a particular value in an activity. The converted runbook converts the activity and specifies a variable asset in Azure Automation with the same name as the Orchestrator variable. This action is noted in the **Runbook Converter - Summary.log** file that is created after the conversion. You must manually create this variable asset in Azure Automation before using the runbook.

Work with Orchestrator input parameters

Runbooks in Orchestrator accept input parameters with the `Initialize Data` activity. If the runbook being converted includes this activity, then an [input parameter](#) in the Azure Automation runbook is created for each parameter in the activity. A [Workflow Script control](#) activity is created in the converted runbook that retrieves and returns each parameter. Any activities in the runbook that use an input parameter refer to the output from this activity.

The reason that this strategy is used is to best mirror the functionality in the Orchestrator runbook. Activities in new graphical runbooks should refer directly to input parameters using a Runbook input data source.

Invoke Runbook activity

Runbooks in Orchestrator start other runbooks with the `Invoke Runbook` activity. If the runbook being converted includes this activity and the `Wait for completion` option is set, then a runbook activity is created for it in the converted runbook. If the `Wait for completion` option is not set, then a Workflow Script activity is created that uses [Start-AzAutomationRunbook](#) to start the runbook. After you import the converted runbook into Azure Automation, you must modify this activity with the information specified in the activity.

Create Orchestrator assets

The Runbook Converter does not convert Orchestrator assets. You must manually create any required Orchestrator assets in Azure Automation.

Configure Hybrid Runbook Worker

Orchestrator stores runbooks on a database server and runs them on runbook servers, both in your local datacenter. Runbooks in Azure Automation are stored in the Azure cloud and can run in your local datacenter using a [Hybrid Runbook Worker](#). Configure a worker to run your runbooks converted from Orchestrator, since they are designed to run on local servers and access local resources.

Related articles

- For Orchestrator details, see [System Center 2012 - Orchestrator](#).
- Learn more about automating the management of services in [Service Management Automation](#).
- Details of Orchestrator activities can be found in [Orchestrator Standard Activities](#).
- To obtain the Orchestrator migration toolkit, see [Download System Center Orchestrator Migration Toolkit](#).
- For an overview of the Azure Automation Hybrid Runbook Worker, see [Hybrid Runbook Worker overview](#).

Manage certificates in Azure Automation

4/22/2021 • 4 minutes to read • [Edit Online](#)

Azure Automation stores certificates securely for access by runbooks and DSC configurations, by using the [Get-AzAutomationCertificate](#) cmdlet for Azure Resource Manager resources. Secure certificate storage allows you to create runbooks and DSC configurations that use certificates for authentication, or add them to Azure or third-party resources.

NOTE

Secure assets in Azure Automation include credentials, certificates, connections, and encrypted variables. These assets are encrypted and stored in Automation by using a unique key that is generated for each Automation account. Automation stores the key in the system-managed Key Vault service. Before storing a secure asset, Automation loads the key from Key Vault, and then uses it to encrypt the asset.

PowerShell cmdlets to access certificates

The cmdlets in the following table create and manage Automation certificates with PowerShell. They ship as part of the [Az modules](#).

CMDLET	DESCRIPTION
Get-AzAutomationCertificate	Retrieves information about a certificate to use in a runbook or DSC configuration. You can only retrieve the certificate itself by using the internal <code>Get-AutomationCertificate</code> cmdlet.
New-AzAutomationCertificate	Creates a new certificate in Automation.
Remove-AzAutomationCertificate	Removes a certificate from Automation.
Set-AzAutomationCertificate	Sets the properties for an existing certificate, including uploading the certificate file and setting the password for a .pfx file.

The [Add-AzureCertificate](#) cmdlet can also be used to upload a service certificate for the specified cloud service.

Internal cmdlets to access certificates

The internal cmdlet in the following table is used to access certificates in your runbooks. This cmdlet comes with the global module `Orchestrator.AssetManagement.Cmdlets`. For more information, see [Internal cmdlets](#).

INTERNAL CMDLET	DESCRIPTION
<code>Get-AutomationCertificate</code>	Gets a certificate to use in a runbook or DSC configuration. Returns a <code>System.Security.Cryptography.X509Certificates.X509Certificate2</code> object.

NOTE

You should avoid using variables in the `Name` parameter of `Get-AutomationCertificate` in a runbook or DSC configuration. Such variables can complicate discovery of dependencies between runbooks or DSC configurations and Automation variables at design time.

Python functions to access certificates

Use the function in the following table to access certificates in a Python 2 and 3 runbook. Python 3 runbooks are currently in preview.

FUNCTION	DESCRIPTION
<code>automationassets.get_automation_certificate</code>	Retrieves information about a certificate asset.

NOTE

You must import the `automationassets` module at the beginning of your Python runbook to access the asset functions.

Create a new certificate

When you create a new certificate, you upload a .cer or .pfx file to Automation. If you mark the certificate as exportable, then you can transfer it out of the Automation certificate store. If it isn't exportable, then it can only be used for signing within the runbook or DSC configuration. Automation requires the certificate to have the provider **Microsoft Enhanced RSA and AES Cryptographic Provider**.

Create a new certificate with the Azure portal

1. From your Automation account, on the left-hand pane select **Certificates** under **Shared Resource**.
2. On the **Certificates** page, select **Add a certificate**.
3. In the **Name** field, type a name for the certificate.
4. To browse for a .cer or .pfx file, under **Upload a certificate file**, choose **Select a file**. If you select a .pfx file, specify a password and indicate if it can be exported.
5. Select **Create** to save the new certificate asset.

Create a new certificate with PowerShell

The following example demonstrates how to create a new Automation certificate and mark it exportable. This example imports an existing .pfx file.

```
$certificateName = 'MyCertificate'
$PfxCertPath = '.\MyCert.pfx'
$CertificatePassword = ConvertTo-SecureString -String 'P@$$w0rd' -AsPlainText -Force
$ResourceGroup = "ResourceGroup01"

New-AzAutomationCertificate -AutomationAccountName "MyAutomationAccount" -Name $certificateName -Path
$PfxCertPath -Password $CertificatePassword -Exportable -ResourceGroupName $ResourceGroup
```

Create a new certificate with a Resource Manager template

The following example demonstrates how to deploy a certificate to your Automation account by using a Resource Manager template through PowerShell:

```

$AutomationAccountName = "<automation account name>"
$PfxCertPath = '<PFX cert path and filename>'
$CertificatePassword = '<password>'
$certificateName = '<certificate name>' #A name of your choosing
$ResourceGroupName = '<resource group name>' #The one that holds your automation account
$flags = [System.Security.Cryptography.X509Certificates.X509KeyStorageFlags]::Exportable ` 
    -bor [System.Security.Cryptography.X509Certificates.X509KeyStorageFlags]::PersistKeySet ` 
    -bor [System.Security.Cryptography.X509Certificates.X509KeyStorageFlags]::MachineKeySet
# Load the certificate into memory
$PfxCert = New-Object -TypeName System.Security.Cryptography.X509Certificates.X509Certificate2 -ArgumentList
@($PfxCertPath, $CertificatePassword, $flags)
# Export the certificate and convert into base 64 string
$Base64Value =
[System.Convert]::ToBase64String($PfxCert.Export([System.Security.Cryptography.X509Certificates.X509ContentType]::Pkcs12))
$Thumbprint = $PfxCert.Thumbprint

$json = @"
{
    '$schema': 'https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#',
    'contentVersion': '1.0.0.0',
    'resources': [
        {
            'name': '$AutomationAccountName/$certificateName',
            'type': 'Microsoft.Automation/automationAccounts/certificates',
            'apiVersion': '2015-10-31',
            'properties': {
                'base64Value': '$Base64Value',
                'thumbprint': '$Thumbprint',
                'isExportable': true
            }
        }
    ]
}
"@

$json | out-file .\template.json
New-AzResourceGroupDeployment -Name NewCert -ResourceGroupName $ResourceGroupName -TemplateFile
.\template.json

```

Get a certificate

To retrieve a certificate, use the internal `Get-AutomationCertificate` cmdlet. You can't use the [Get-AzAutomationCertificate](#) cmdlet, because it returns information about the certificate asset, but not the certificate itself.

Textual runbook examples

- [PowerShell](#)
- [Python 2](#)
- [Python 3](#)

The following example shows how to add a certificate to a cloud service in a runbook. In this sample, the password is retrieved from an encrypted automation variable.

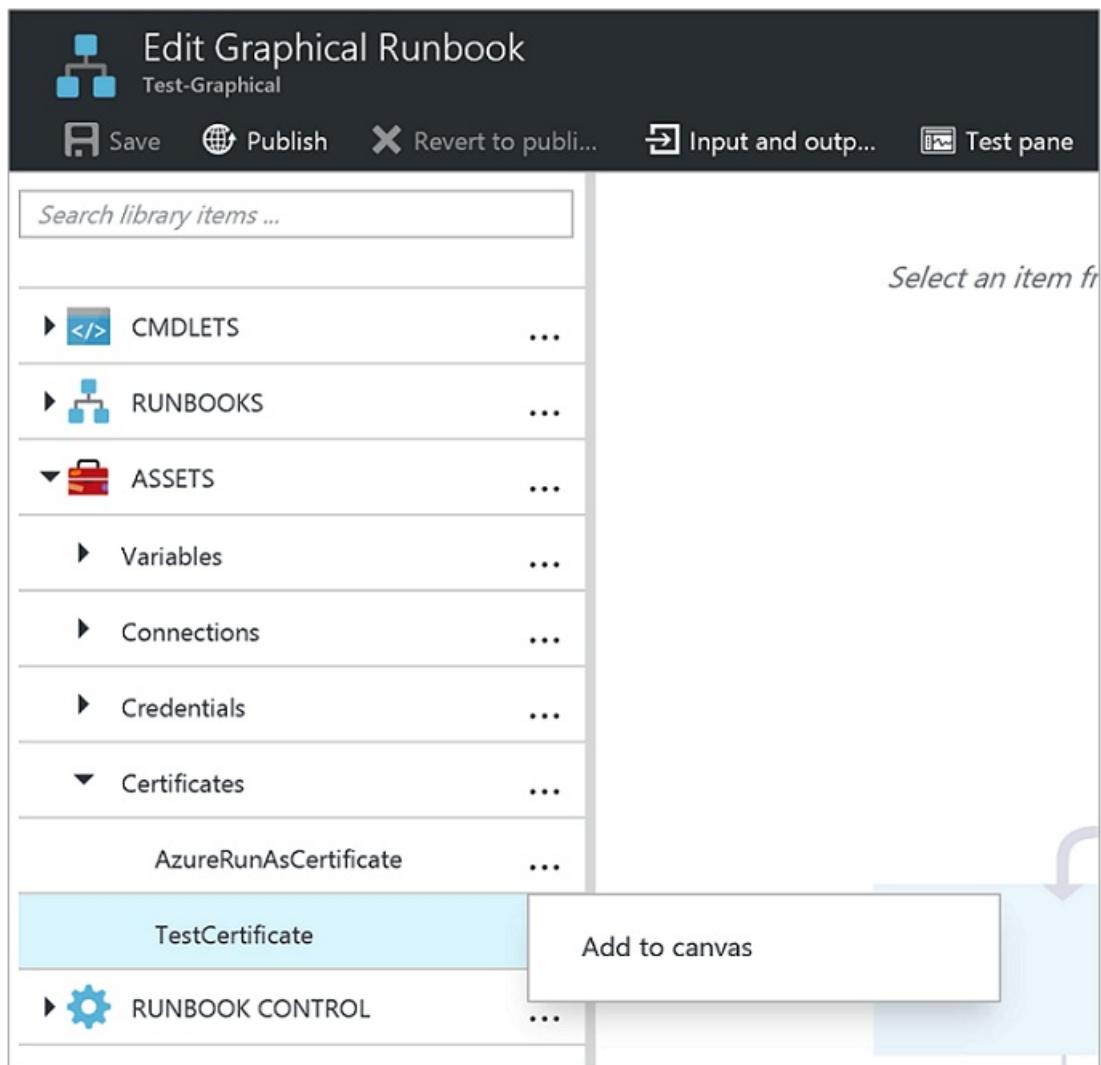
```

$serviceName = 'MyCloudService'
$cert = Get-AutomationCertificate -Name 'MyCertificate'
$certPwd = Get-AzAutomationVariable -ResourceGroupName "ResourceGroup01" ` 
    -AutomationAccountName "MyAutomationAccount" -Name 'MyCertPassword'
Add-AzureCertificate -ServiceName $serviceName -CertToDeploy $cert

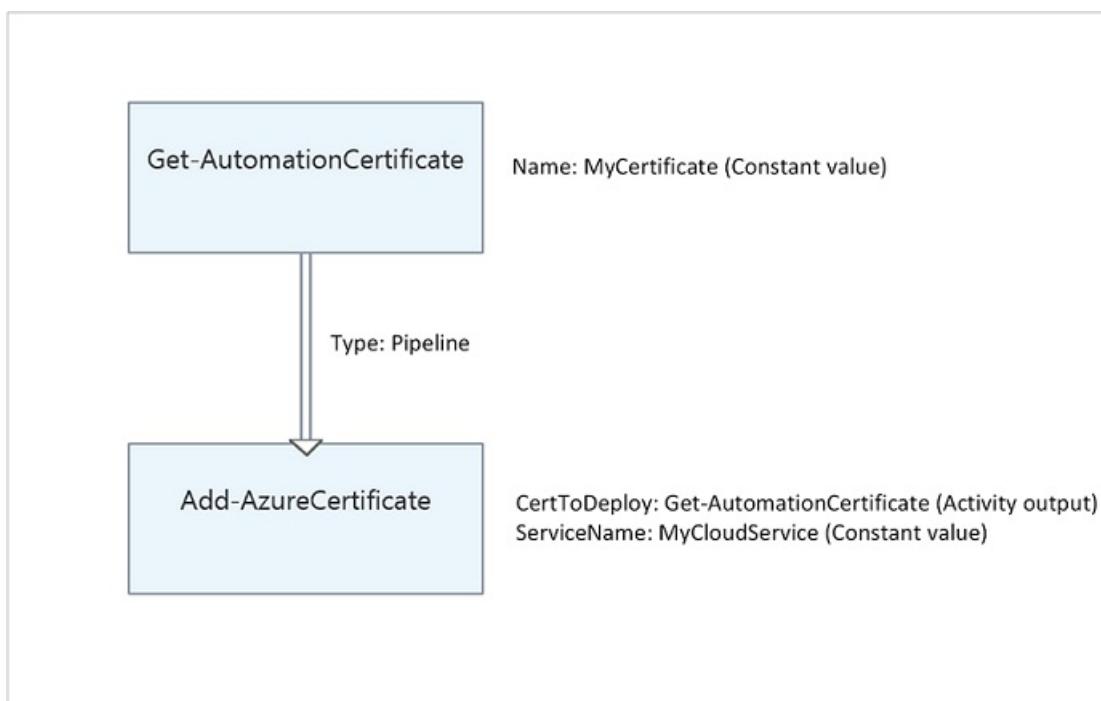
```

Graphical runbook example

Add an activity for the internal `Get-AutomationCertificate` cmdlet to a graphical runbook by right-clicking on the certificate in the Library pane, and selecting **Add to canvas**.



The following image shows an example of using a certificate in a graphical runbook.



Next steps

- To learn more about the cmdlets used to access certificates, see [Manage modules in Azure Automation](#).
- For general information about runbooks, see [Runbook execution in Azure Automation](#).
- For details of DSC configurations, see [Azure Automation State Configuration overview](#).

Manage connections in Azure Automation

8/18/2021 • 6 minutes to read • [Edit Online](#)

An Azure Automation connection asset contains the information listed below. This information is required for connection to an external service or application from a runbook or DSC configuration.

- Information needed for authentication, such as user name and password
- Connection information, such as URL or port

The connection asset keeps together all properties for connecting to a particular application, making it unnecessary to create multiple variables. You can edit the values for a connection in one place, and you can pass the name of a connection to a runbook or DSC configuration in a single parameter. The runbook or configuration accesses the properties for a connection using the internal `Get-AutomationConnection` cmdlet.

When you create a connection, you must specify a connection type. The connection type is a template that defines a set of properties. You can add a connection type to Azure Automation using an integration module with a metadata file. It's also possible to create a connection type using the [Azure Automation API](#) if the integration module includes a connection type and is imported into your Automation account.

NOTE

Secure assets in Azure Automation include credentials, certificates, connections, and encrypted variables. These assets are encrypted and stored in Azure Automation using a unique key that is generated for each Automation account. Azure Automation stores the key in the system-managed Key Vault. Before storing a secure asset, Automation loads the key from Key Vault and then uses it to encrypt the asset.

Connection types

Azure Automation makes the following built-in connection types available:

- `Azure` - Represents a connection used to manage classic resources.
- `AzureServicePrincipal` - Represents a connection used by the Azure Run As account.
- `AzureClassicCertificate` - Represents a connection used by the classic Azure Run As account.

In most cases, you don't need to create a connection resource because it is created when you create a [Run As account](#).

PowerShell cmdlets to access connections

The cmdlets in the following table create and manage Automation connections with PowerShell. They ship as part of the [Az modules](#).

CMDLET	DESCRIPTION
Get-AzAutomationConnection	Retrieves information about a connection.
New-AzAutomationConnection	Creates a new connection.
Remove-AzAutomationConnection	Removes an existing connection.

CMDLET	DESCRIPTION
<code>Set-AzAutomationConnectionFieldValue</code>	Sets the value of a particular field for an existing connection.

Internal cmdlets to access connections

The internal cmdlet in the following table is used to access connections in your runbooks and DSC configurations. This cmdlet comes with the global module `Orchestrator.AssetManagement.Cmdlets`. For more information, see [Internal cmdlets](#).

INTERNAL CMDLET	DESCRIPTION
<code>Get-AutomationConnection</code>	Retrieves the values of the different fields in the connection and returns them as a hashtable . You can then use this hashtable with the appropriate commands in the runbook or DSC configuration.

NOTE

Avoid using variables with the `Name` parameter of `Get-AutomationConnection`. Use of variables in this case can complicate discovery of dependencies between runbooks or DSC configurations and connection assets at design time.

Python functions to access connections

The function in the following table is used to access connections in a Python 2 and 3 runbook. Python 3 runbooks are currently in preview.

FUNCTION	DESCRIPTION
<code>automationassets.get_automation_connection</code>	Retrieves a connection. Returns a dictionary with the properties of the connection.

NOTE

You must import the `automationassets` module at the top of your Python runbook to access the asset functions.

Create a new connection

Create a new connection with the Azure portal

To create a new connection in the Azure portal:

1. From your Automation account, click **Connections** under **Shared Resources**.
2. Click **+ Add a connection** on the Connections page.
3. In the **Type** field on the New Connection pane, select the type of connection to create. Your choices are `Azure`, `AzureServicePrincipal`, and `AzureClassicCertificate`.
4. The form presents properties for the connection type that you've chosen. Complete the form and click **Create** to save the new connection.

Create a new connection with Windows PowerShell

Create a new connection with Windows PowerShell using the `New-AzAutomationConnection` cmdlet. This cmdlet has a `ConnectionFieldValues` parameter that expects a hashtable defining values for each of the properties

defined by the connection type.

You can use the following example commands as an alternative to creating the Run As account from the portal to create a new connection asset.

```
$ConnectionAssetName = "AzureRunAsConnection"
$ConnectionFieldValues = @{"ApplicationId" = $Application.ApplicationId; "TenantId" = $TenantID.TenantId;
"CertificateThumbprint" = $Cert.Thumbprint; "SubscriptionId" = $SubscriptionId}
New-AzAutomationConnection -ResourceGroupName $ResourceGroup -AutomationAccountName $AutomationAccountName -
Name $ConnectionAssetName -ConnectionType AzureServicePrincipal -ConnectionFieldValues
$ConnectionFieldValues
```

When you create your Automation account, it includes several global modules by default, along with the connection type `AzureServicePrincipal` to create the `AzureRunAsConnection` connection asset. If you try to create a new connection asset to connect to a service or application with a different authentication method, the operation fails because the connection type is not already defined in your Automation account. For more information on creating your own connection type for a custom module, see [Add a connection type](#).

Add a connection type

If your runbook or DSC configuration connects to an external service, you must define a connection type in a [custom module](#) called an integration module. This module includes a metadata file that specifies connection type properties and is named `<ModuleName>-Automation.json`, located in the module folder of your compressed `.zip` file. This file contains the fields of a connection that are required to connect to the system or service that the module represents. Using this file, you can set the field names, data types, encryption status, and optional status for the connection type.

The following example is a template in the `.json` file format that defines user name and password properties for a custom connection type called `MyModuleConnection`:

```
{
  "ConnectionFields": [
    {
      "IsEncrypted": false,
      "IsOptional": true,
      "Name": "Username",
      "TypeName": "System.String"
    },
    {
      "IsEncrypted": true,
      "IsOptional": false,
      "Name": "Password",
      "TypeName": "System.String"
    }
  ],
  "ConnectionTypeName": "MyModuleConnection",
  "IntegrationModuleName": "MyModule"
}
```

Get a connection in a runbook or DSC configuration

Retrieve a connection in a runbook or DSC configuration with the internal `Get-AutomationConnection` cmdlet.

This cmdlet is preferred over the `Get-AzAutomationConnection` cmdlet, as it retrieves the connection values instead of information about the connection.

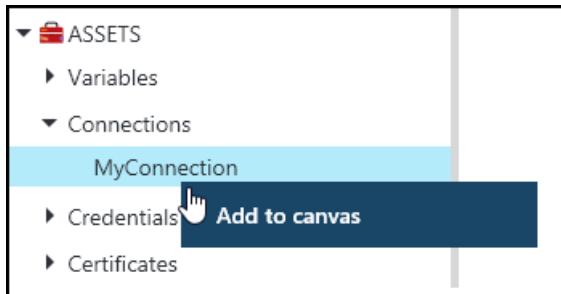
- [PowerShell](#)
- [Python](#)

The following example shows how to use the Run As account to authenticate with Azure Resource Manager resources in your runbook. It uses a connection asset representing the Run As account, which references the certificate-based service principal.

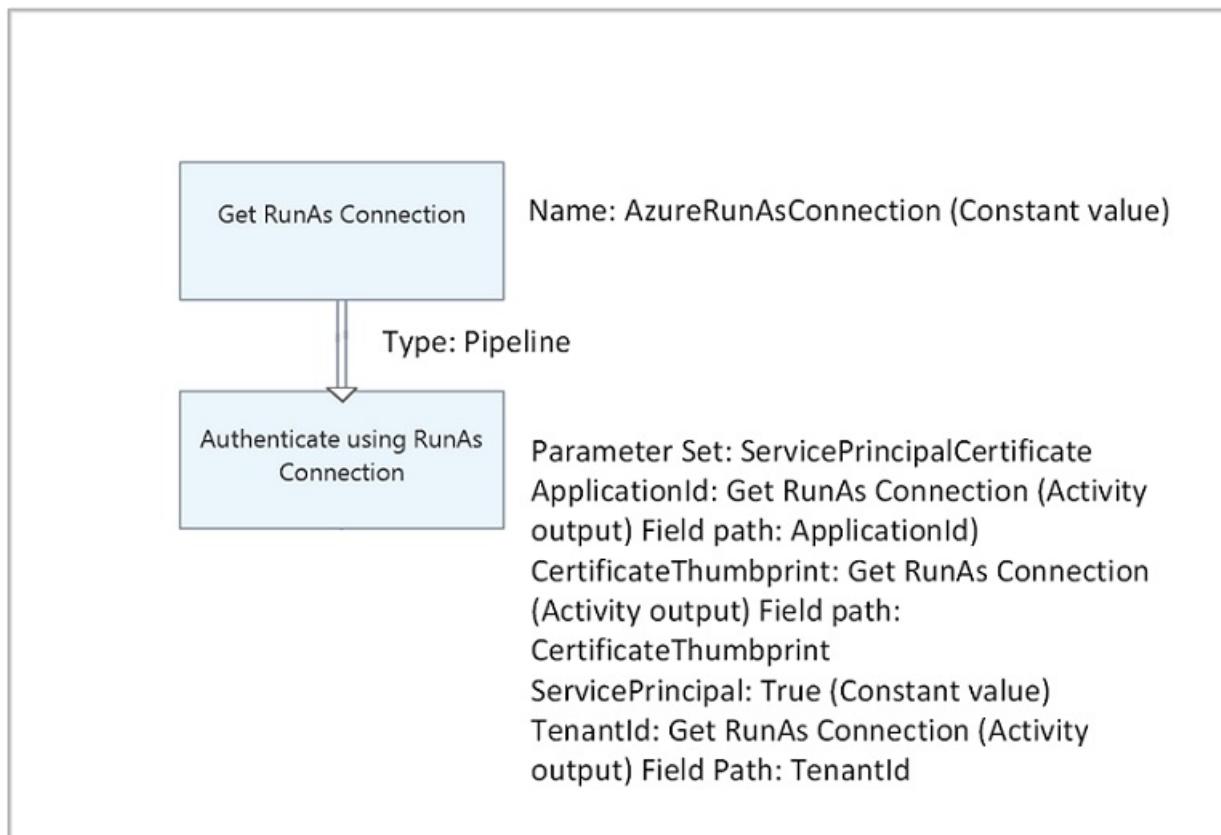
```
$Conn = Get-AutomationConnection -Name AzureRunAsConnection  
Connect-AzAccount -ServicePrincipal -Tenant $Conn.TenantID -ApplicationId $Conn.ApplicationID -  
CertificateThumbprint $Conn.CertificateThumbprint
```

Graphical runbook examples

You can add an activity for the internal `Get-AutomationConnection` cmdlet to a graphical runbook. Right-click the connection in the Library pane of the graphical editor and select **Add to canvas**.



The following image shows an example of using a connection object in a graphical runbook. This example uses the `Constant value` data set for the `Get RunAs Connection` activity, which uses a connection object for authentication. A **pipeline link** is used here since the `ServicePrincipalCertificate` parameter set is expecting a single object.



Next steps

- To learn more about the cmdlets used to access connections, see [Manage modules in Azure Automation](#).
- For general information about runbooks, see [Runbook execution in Azure Automation](#).
- For details of DSC configurations, see [State Configuration overview](#).

Manage credentials in Azure Automation

4/22/2021 • 5 minutes to read • [Edit Online](#)

An Automation credential asset holds an object that contains security credentials, such as a user name and a password. Runbooks and DSC configurations use cmdlets that accept a `PSCredential` object for authentication. Alternatively, they can extract the user name and password of the `PSCredential` object to provide to some application or service requiring authentication.

NOTE

Secure assets in Azure Automation include credentials, certificates, connections, and encrypted variables. These assets are encrypted and stored in Azure Automation using a unique key that is generated for each Automation account. Azure Automation stores the key in the system-managed Key Vault. Before storing a secure asset, Automation loads the key from Key Vault and then uses it to encrypt the asset.

NOTE

For information about viewing or deleting personal data, see [Azure Data Subject Requests for the GDPR](#). For more information about GDPR, see the [GDPR section of the Microsoft Trust Center](#) and the [GDPR section of the Service Trust portal](#).

PowerShell cmdlets used to access credentials

The cmdlets in the following table create and manage Automation credentials with PowerShell. They ship as part of the [Az modules](#).

CMDLET	DESCRIPTION
Get-AzAutomationCredential	Retrieves a <code>CredentialInfo</code> object containing metadata about the credential. The cmdlet doesn't retrieve the <code>PSCredential</code> object itself.
New-AzAutomationCredential	Creates a new Automation credential.
Remove-AzAutomationCredential	Removes an Automation credential.
Set-AzAutomationCredential	Sets the properties for an existing Automation credential.

Other cmdlets used to access credentials

The cmdlets in the following table are used to access credentials in your runbooks and DSC configurations.

CMDLET	DESCRIPTION
--------	-------------

CMDLET	DESCRIPTION
<code>Get-AutomationPSCredential</code>	Gets a <code>PSCredential</code> object to use in a runbook or DSC configuration. Most often, you should use this internal cmdlet instead of the <code>Get-AzAutomationCredential</code> cmdlet, as the latter only retrieves credential information. This information isn't normally helpful to pass to another cmdlet.
<code>Get-Credential</code>	Gets a credential with a prompt for user name and password. This cmdlet is part of the default <code>Microsoft.PowerShell.Security</code> module. See Default modules .
<code>New-AzureAutomationCredential</code>	Creates a credential asset. This cmdlet is part of the default Azure module. See Default modules .

To retrieve `PSCredential` objects in your code, you must import the `Orchestrator.AssetManagement.Cmdlets` module. For more information, see [Manage modules in Azure Automation](#).

```
Import-Module Orchestrator.AssetManagement.Cmdlets -ErrorAction SilentlyContinue
```

NOTE

You should avoid using variables in the `Name` parameter of `Get-AutomationPSCredential`. Their use can complicate discovery of dependencies between runbooks or DSC configurations and credential assets at design time.

Python functions that access credentials

The function in the following table is used to access credentials in a Python 2 and 3 runbook. Python 3 runbooks are currently in preview.

FUNCTION	DESCRIPTION
<code>automationassets.get_automation_credential</code>	Retrieves information about a credential asset.

NOTE

Import the `automationassets` module at the top of your Python runbook to access the asset functions.

Create a new credential asset

You can create a new credential asset using the Azure portal or using Windows PowerShell.

Create a new credential asset with the Azure portal

- From your Automation account, on the left-hand pane select **Credentials** under **Shared Resources**.
- On the **Credentials** page, select **Add a credential**.
- In the New Credential pane, enter an appropriate credential name following your naming standards.
- Type your access ID in the **User name** field.
- For both password fields, enter your secret access key.

New Credential

Name *

Description

User name *

Password *

Confirm password *

6. If the multi-factor authentication box is checked, uncheck it.

7. Click **Create** to save the new credential asset.

NOTE

Azure Automation does not support user accounts that use multi-factor authentication.

Create a new credential asset with Windows PowerShell

The following example shows how to create a new Automation credential asset. A `PSCredential` object is first created with the name and password, and then used to create the credential asset. Instead, you can use the `Get-Credential` cmdlet to prompt the user to type in a name and password.

```
$user = "MyDomain\MyUser"
$pw = ConvertTo-SecureString "PassWord!" -AsPlainText -Force
$cred = New-Object -TypeName System.Management.Automation.PSCredential -ArgumentList $user, $pw
New-AzureAutomationCredential -AutomationAccountName "MyAutomationAccount" -Name "MyCredential" -Value $cred
```

Get a credential asset

A runbook or DSC configuration retrieves a credential asset with the internal `Get-AutomationPSCredential` cmdlet. This cmdlet gets a `PSCredential` object that you can use with a cmdlet that requires a credential. You can also retrieve the properties of the credential object to use individually. The object has properties for the user name and the secure password.

NOTE

The `Get-AzAutomationCredential` cmdlet does not retrieve a `PSCredential` object that can be used for authentication. It only provides information about the credential. If you need to use a credential in a runbook, you must retrieve it as a `PSCredential` object using `Get-AutomationPSCredential`.

Alternatively, you can use the `GetNetworkCredential` method to retrieve a `NetworkCredential` object that represents an unsecured version of the password.

Textual runbook example

- [PowerShell](#)
- [Python 2](#)
- [Python 3](#)

The following example shows how to use a PowerShell credential in a runbook. It retrieves the credential and assigns its user name and password to variables.

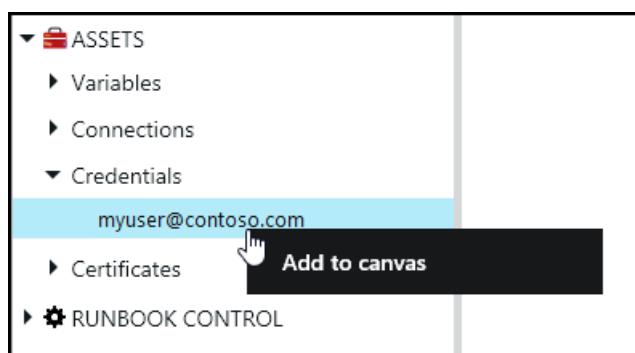
```
$myCredential = Get-AutomationPSCredential -Name 'MyCredential'  
$userName = $myCredential.UserName  
$securePassword = $myCredential.Password  
$password = $myCredential.GetNetworkCredential().Password
```

You can also use a credential to authenticate to Azure with [Connect-AzAccount](#). Under most circumstances, you should use a [Run As account](#) and retrieve the connection with [Get-AzAutomationConnection](#).

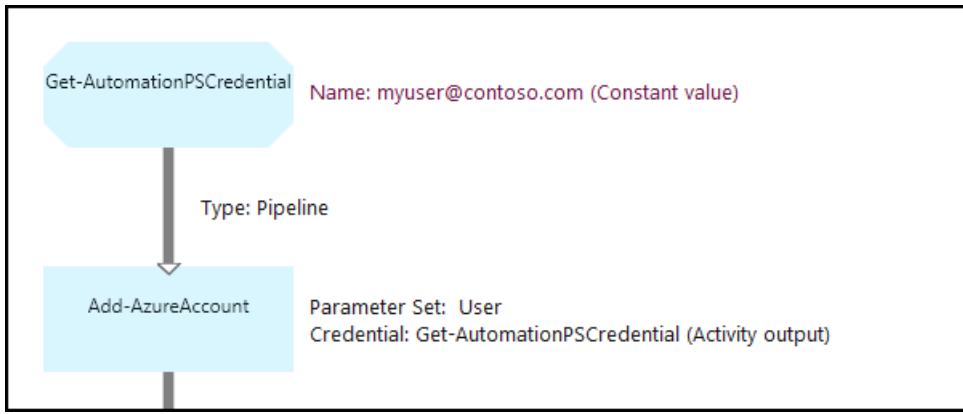
```
$myCred = Get-AutomationPSCredential -Name 'MyCredential'  
$userName = $myCred.UserName  
$securePassword = $myCred.Password  
$password = $myCred.GetNetworkCredential().Password  
  
$myPsCred = New-Object System.Management.Automation.PSCredential ($userName,$securePassword)  
  
Connect-AzAccount -Credential $myPsCred
```

Graphical runbook example

You can add an activity for the internal `Get-AutomationPSCredential` cmdlet to a graphical runbook by right-clicking on the credential in the Library pane of the graphical editor and selecting **Add to canvas**.



The following image shows an example of using a credential in a graphical runbook. In this case, the credential provides authentication for a runbook to Azure resources, as described in [Use Azure AD in Azure Automation to authenticate to Azure](#). The first activity retrieves the credential that has access to the Azure subscription. The account connection activity then uses this credential to provide authentication for any activities that come after it. A [pipeline link](#) is used here since `Get-AutomationPSCredential` is expecting a single object.



Use credentials in a DSC configuration

While DSC configurations in Azure Automation can work with credential assets using `Get-AutomationPSCredential`, they can also pass credential assets via parameters. For more information, see [Compiling configurations in Azure Automation DSC](#).

Next steps

- To learn more about the cmdlets used to access certificates, see [Manage modules in Azure Automation](#).
- For general information about runbooks, see [Runbook execution in Azure Automation](#).
- For details of DSC configurations, see [Azure Automation State Configuration overview](#).

Manage modules in Azure Automation

7/21/2021 • 16 minutes to read • [Edit Online](#)

Azure Automation uses a number of PowerShell modules to enable cmdlets in runbooks and DSC resources in DSC configurations. Supported modules include:

- [Azure PowerShell Az.Automation](#).
- [Azure PowerShell AzureRM.Automation](#).
- Other PowerShell modules.
- Internal `Orchestrator.AssetManagement.Cmdlets` module.
- Python 2 modules.
- Custom modules that you create.

When you create an Automation account, Azure Automation imports some modules by default. See [Default modules](#).

Sandboxes

When Automation executes runbook and DSC compilation jobs, it loads the modules into sandboxes where the runbooks can run and the DSC configurations can compile. Automation also automatically places any DSC resources in modules on the DSC pull server. Machines can pull the resources when they apply the DSC configurations.

NOTE

Be sure to import only the modules that your runbooks and DSC configurations require. We don't recommend importing the root Az module. It includes many other modules that you might not need, which can cause performance problems. Import individual modules, such as Az.Compute, instead.

Cloud sandbox supports a maximum of 48 system calls, and restricts all other calls for security reasons. Other functionality such as credential management and some networking is not supported in the cloud sandbox.

Due to the number of modules and cmdlets included, it's difficult to know beforehand which of the cmdlets will make unsupported calls. Generally, we have seen issues with cmdlets which require elevated access, require a credential as a parameter, or cmdlets related to networking. Any cmdlets that perform full stack network operations are not supported in the sandbox, including [Connect-AipService](#) from the AIPService PowerShell module and [Resolve-DnsName](#) from the DNSClient module.

These are known limitations with the sandbox. The recommended workaround is to deploy a [Hybrid Runbook Worker](#) or use [Azure Functions](#).

IMPORTANT

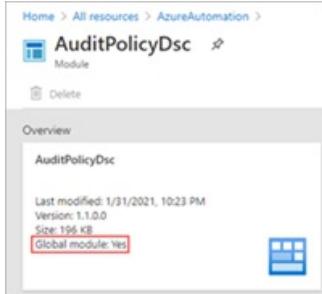
Do not include the keyword "AzureRm" in any script designed to be executed with the Az module. Inclusion of the keyword, even in a comment, may cause the AzureRm to load and then conflict with the Az module.

Default modules

The following table lists modules that Azure Automation imports by default when you create your Automation

account. Automation can import newer versions of these modules. However, you can't remove the original version from your Automation account, even if you delete a newer version. Note that these default modules include several AzureRM modules.

The default modules are also known as global modules. In the Azure portal, the **Global module** property will be **true** when viewing a module that was imported when the account was created.



Automation doesn't import the root Az module automatically into any new or existing Automation accounts. For more about working with these modules, see [Migrating to Az modules](#).

NOTE

We don't recommend altering modules and runbooks in Automation accounts used for deployment of the [Start/Stop VMs during off-hours](#) feature.

MODULE NAME	VERSION
AuditPolicyDsc	1.1.0.0
Azure	1.0.3
Azure.Storage	1.0.3
AzureRM.Automation	1.0.3
AzureRM.Compute	1.2.1
AzureRM.Profile	1.0.3
AzureRM.Resources	1.0.3
AzureRM.Sql	1.0.3
AzureRM.Storage	1.0.3
ComputerManagementDsc	5.0.0.0
GPRRegistryPolicyParser	0.2
Microsoft.PowerShell.Core	0
Microsoft.PowerShell.Diagnostics	
Microsoft.PowerShell.Management	

MODULE NAME	VERSION
Microsoft.PowerShell.Security	
Microsoft.PowerShell.Utility	
Microsoft.WSMan.Management	
Orchestrator.AssetManagement.Cmdlets	1
PSDscResources	2.9.0.0
SecurityPolicyDsc	2.1.0.0
StateConfigCompositeResources	1
xDSCDomainjoin	1.1
xPowerShellExecutionPolicy	1.1.0.0
xRemoteDesktopAdmin	1.1.0.0

Az modules

For Az.Automation, the majority of the cmdlets have the same names as those used for the AzureRM modules, except that the `AzureRM` prefix has been changed to `Az`. For a list of Az modules that don't follow this naming convention, see the [list of exceptions](#).

Internal cmdlets

Azure Automation supports internal cmdlets that are only available when you execute runbooks in the Azure sandbox environment or on a Windows Hybrid Runbook Worker. The internal module `Orchestrator.AssetManagement.Cmdlets` is installed by default in your Automation account and when the Windows Hybrid Runbook Worker role is installed on the machine.

The following table defines the internal cmdlets. These cmdlets are designed to be used instead of Azure PowerShell cmdlets to interact with your Automation account resources. They can retrieve secrets from encrypted variables, credentials, and encrypted connections.

NAME	DESCRIPTION
Get-AutomationCertificate	<code>Get-AutomationCertificate [-Name] <string> [<CommonParameters>]</code>
Get-AutomationConnection	<code>Get-AutomationConnection [-Name] <string> [-DoNotDecrypt] [<CommonParameters>]</code>
Get-AutomationPSCredential	<code>Get-AutomationPSCredential [-Name] <string> [<CommonParameters>]</code>
Get-AutomationVariable	<code>Get-AutomationVariable [-Name] <string> [-DoNotDecrypt] [<CommonParameters>]</code>

NAME	DESCRIPTION
Set-AutomationVariable	<code>Set-AutomationVariable [-Name] <string> -Value <Object> [<CommonParameters>]</code>
Start-AutomationRunbook	<code>Start-AutomationRunbook [-Name] <string> [-Parameters <IDictionary>] [-RunOn <string>] [-JobId <guid>] [<CommonParameters>]</code>
Wait-AutomationJob	<code>Wait-AutomationJob -Id <guid[]> [-TimeoutInMinutes <int>] [-DelayInSeconds <int>] [-OutputJobsTransitionedToRunning] [<CommonParameters>]</code>

Note that the internal cmdlets differ in naming from the Az and AzureRM cmdlets. Internal cmdlet names don't contain words like `Azure` or `Az` in the noun, but do use the word `Automation`. We recommend their use over the use of Az or AzureRM cmdlets during runbook execution in an Azure sandbox or on a Windows Hybrid Runbook Worker because they require fewer parameters and run in the context of your job during execution.

Use Az or AzureRM cmdlets for manipulating Automation resources outside the context of a runbook.

Python modules

You can create Python 2 runbooks in Azure Automation. For Python module information, see [Manage Python 2 packages in Azure Automation](#).

Custom modules

Azure Automation supports custom PowerShell modules that you create to use with your runbooks and DSC configurations. One type of custom module is an integration module that optionally contains a file of metadata to define the custom functionality for the module cmdlets. An example of the use of an integration module is provided in [Add a connection type](#).

Azure Automation can import a custom module to make its cmdlets available. Behind the scenes, it stores the module and uses it in the Azure sandboxes, just like it does other modules.

Migrate to Az modules

This section tells how to migrate to the Az modules in Automation. For more information, see [Migrate Azure PowerShell from AzureRM to Az](#).

We don't recommend running AzureRM modules and Az modules in the same Automation account. When you're sure you want to migrate from AzureRM to Az, it's best to fully commit to a complete migration. Automation often reuses sandboxes within the Automation account to save on startup times. If you don't make a full module migration, you might start a job that uses only AzureRM modules, and then start another job that uses only Az modules. The sandbox soon crashes, and you receive an error stating that the modules aren't compatible. This situation results in randomly occurring crashes for any particular runbook or configuration.

NOTE

When you create a new Automation account, even after migration to Az modules, Automation installs the AzureRM modules by default. You can still update the tutorial runbooks with the AzureRM cmdlets. However, you shouldn't run these runbooks.

Test your runbooks and DSC configurations prior to module migration

Be sure to test all runbooks and DSC configurations carefully, in a separate Automation account, before

migrating to the Az modules.

Stop and unschedule all runbooks that use AzureRM modules

To ensure that you don't run any existing runbooks or DSC configurations that use AzureRM modules, you must stop and unschedule all affected runbooks and configurations. First, make sure that you review each runbook or DSC configuration and its schedules separately, to ensure that you can reschedule the item in the future if necessary.

When you're ready to remove your schedules, you can either use the Azure portal or the [Remove-AzureRmAutomationSchedule](#) cmdlet. See [Remove a schedule](#).

Remove AzureRM modules

It's possible to remove the AzureRM modules before you import the Az modules. However, if you do, you can interrupt source control synchronization and cause any scripts that are still scheduled to fail. If you decide to remove the modules, see [Uninstall AzureRM](#).

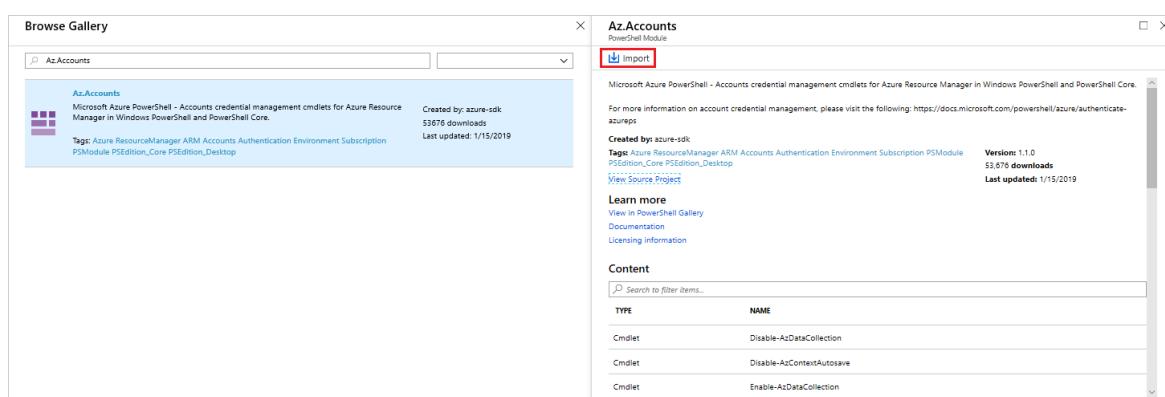
Import Az modules

Importing an Az module into your Automation account doesn't automatically import the module into the PowerShell session that runbooks use. Modules are imported into the PowerShell session in the following situations:

- When a runbook invokes a cmdlet from a module.
- When a runbook imports the module explicitly with the [Import-Module](#) cmdlet.
- When a runbook imports the module explicitly with the [using module](#) statement. The using statement is supported starting with Windows PowerShell 5.0 and supports classes and enum type import.
- When a runbook imports another dependent module.

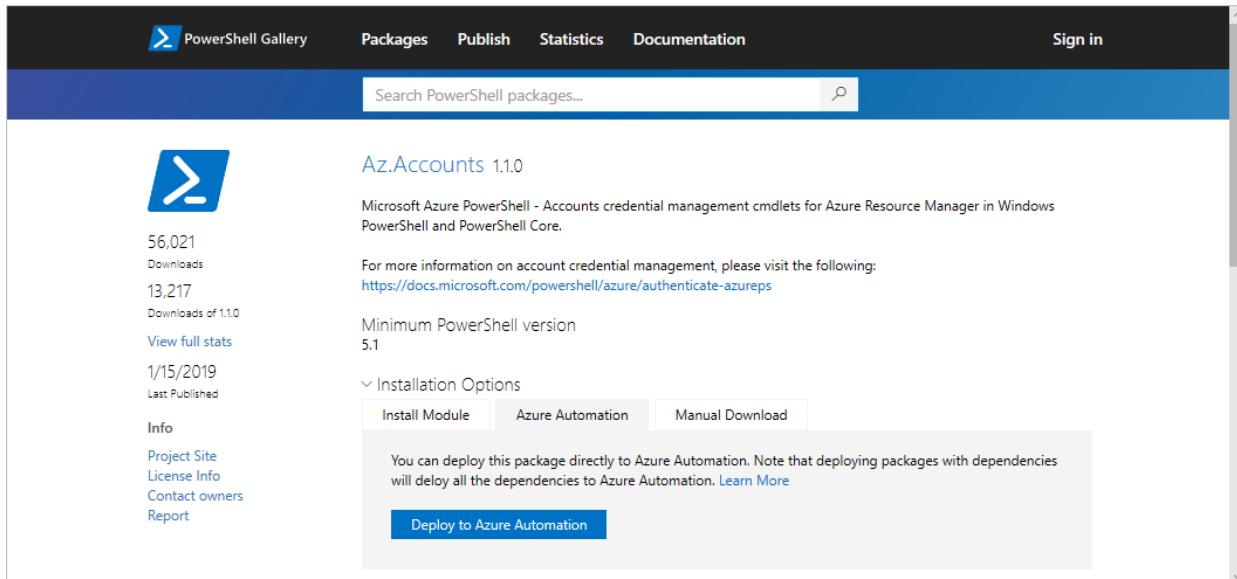
You can import the Az modules into the Automation account from the Azure portal. Remember to import only the Az modules that you need, not every Az module that's available. Because [Az.Accounts](#) is a dependency for the other Az modules, be sure to import this module before any others.

1. Sign in to the Azure [portal](#).
2. Search for and select **Automation Accounts**.
3. On the **Automation Accounts** page, select your Automation account from the list.
4. From your Automation account, under **Shared Resources**, select **Modules**.
5. Select **Browse Gallery**.
6. In the search bar, enter the module name (for example, `Az.Accounts`).
7. On the PowerShell Module page, select **Import** to import the module into your Automation account.



You can also do this import through the [PowerShell Gallery](#), by searching for the module to import. When you

find the module, select it, and choose the **Azure Automation** tab. Select **Deploy to Azure Automation**.



The screenshot shows the PowerShell Gallery interface. At the top, there are navigation links for 'PowerShell Gallery', 'Packages', 'Publish', 'Statistics', 'Documentation', and 'Sign in'. Below the navigation is a search bar with the placeholder 'Search PowerShell packages...'. The main content area displays the 'Az.Accounts' package version 1.1.0. To the left is a sidebar with download statistics: 56,021 total downloads and 13,217 since version 1.1.0. It also includes links for 'View full stats', 'Last Published' (1/15/2019), and 'Info' (Project Site, License Info, Contact owners, Report). The main content area shows the package summary: 'Microsoft Azure PowerShell - Accounts credential management cmdlets for Azure Resource Manager in Windows PowerShell and PowerShell Core.' It provides a link for more information on account credential management (<https://docs.microsoft.com/powershell/azure/authenticate-azurmps>). It specifies a 'Minimum PowerShell version' of 5.1. Below this is a section titled 'Installation Options' with three buttons: 'Install Module' (selected), 'Azure Automation' (highlighted in blue), and 'Manual Download'. A note states: 'You can deploy this package directly to Azure Automation. Note that deploying packages with dependencies will delay all the dependencies to Azure Automation.' A 'Learn More' link is provided. At the bottom of this section is a blue 'Deploy to Azure Automation' button.

Test your runbooks

After you've imported the Az modules into the Automation account, you can start editing your runbooks and DSC configurations to use the new modules. One way to test the modification of a runbook to use the new cmdlets is by using the `Enable-AzureRmAlias -Scope Process` command at the beginning of the runbook. By adding this command to your runbook, the script can run without changes.

Author modules

We recommend that you follow the considerations in this section when you author a custom PowerShell module for use in Azure Automation. To prepare your module for import, you must create at least a .psd1, .psm1, or PowerShell module .dll file with the same name as the module folder. Then you zip up the module folder so that Azure Automation can import it as a single file. The .zip package should have the same name as the contained module folder.

To learn more about authoring a PowerShell module, see [How to Write a PowerShell Script Module](#).

Version folder

PowerShell side-by-side module versioning allows you to use more than one version of a module within PowerShell. This can be useful if you have older scripts that have been tested and only work against a certain version of a PowerShell module, but other scripts require a newer version of the same PowerShell module.

To construct PowerShell modules so they contain multiple versions, create the module folder and then create a folder within this module folder for each version of the module you want to be usable. In the following example, a module called *TestModule* provides two versions, 1.0.0 and 2.0.0.

```
TestModule  
1.0.0  
2.0.0
```

Within each of the version folders, copy your PowerShell .psm1, .psd1, or PowerShell module .dll files that make up a module into the respective version folder. Zip up the module folder so that Azure Automation can import it as a single .zip file. While Automation only shows the highest version of the module imported, if the module package contains side-by-side versions of the module, they are all available for use in your runbooks or DSC configurations.

While Automation supports modules containing side-by-side versions within the same package, it does not support using multiple versions of a module across module package imports. For example, you import module

A, which contains versions 1 and 2 into your Automation account. Later you update **module A** to include versions 3 and 4, when you import into your Automation account, only versions 3 and 4 are usable within any runbooks or DSC configurations. If you require all versions - 1, 2, 3, and 4 to be available, the .zip file your import should contain versions 1, 2, 3, and 4.

If you're going to use different versions of the same module between runbooks, you should always declare the version you want to use in your runbook using the `Import-Module` cmdlet and include the parameter `-RequiredVersion <version>`. Even if the version you want to use is the latest version. This is because runbook jobs may run in the same sandbox. If the sandbox has already explicitly loaded a module of a certain version number, because a previous job in that sandbox said to do so, future jobs in that sandbox won't automatically load the latest version of that module. This is because some version of it is already loaded in the sandbox.

For a DSC resource, use the following command to specify a particular version:

```
Import-DscResource -ModuleName <ModuleName> -ModuleVersion <version>
```

Help information

Include a synopsis, description, and help URI for every cmdlet in your module. In PowerShell, you can define help information for cmdlets by using the `Get-Help` cmdlet. The following example shows how to define a synopsis and help URI in a .psm1 module file.

```
<#
    .SYNOPSIS
    Gets a Contoso User account
#>
function Get-ContosoUser {
[CmdletBinding](DefaultParameterSetName='UseConnectionObject',
    HelpUri='https://www.contoso.com/docs/information')
[OutputType([String])]
param(
    [Parameter(ParameterSetName='UserAccount', Mandatory=true)]
    [ValidateNotNullOrEmpty()]
    [string]
    $UserName,
    [Parameter(ParameterSetName='UserAccount', Mandatory=true)]
    [ValidateNotNullOrEmpty()]
    [string]
    $Password,
    [Parameter(ParameterSetName='ConnectionObject', Mandatory=true)]
    [ValidateNotNullOrEmpty()]
    [Hashtable]
    $Connection
)
switch ($PSCmdlet.ParameterSetName) {
    "UserAccount" {
        $cred = New-Object -TypeName System.Management.Automation.PSCredential -ArgumentList $UserName,
$Password
        Connect-Contoso -Credential $cred
    }
    "ConnectionObject" {
        Connect-Contoso -Connection $Connection
    }
}
```

Providing this information shows help text via the `Get-Help` cmdlet in the PowerShell console. This text is also displayed in the Azure portal.

Activities	
13	cube
NAME	DESCRIPTION
Connect-WSMan	Connect-WSMan [[-ComputerName] <string>] [-Appli...]
Disable-WSManCredSSP	Disable-WSManCredSSP [-Role] <string> [<Common...]
Disconnect-WSMan	Disconnect-WSMan [[-ComputerName] <string>] [<C...]
Enable-WSManCredSSP	Enable-WSManCredSSP [-Role] <string> [[-DelegateC...]
Get-WSManCredSSP	Get-WSManCredSSP [<CommonParameters>]
Get-WSManInstance	Get-WSManInstance [-ResourceURI] <uri> [-Appli...
Invoke-WSManAction	Invoke-WSManAction [-ResourceURI] <uri> [-Action]...

Connection type

If the module connects to an external service, define a connection type by using a [custom integration module](#). Each cmdlet in the module should accept an instance of that connection type (connection object) as a parameter. Users map parameters of the connection asset to the cmdlet's corresponding parameters each time they call a cmdlet.

The screenshot shows the Azure portal interface. On the left, there is a navigation sidebar with links like 'Inventory', 'Change tracking', 'State configuration (DSC)', 'Update management', 'Process Automation', and 'Shared Resources'. The main area displays a list of existing connections:

NAME	TYPE
AzureClassicRunAsConnection	AzureClassicCert
AzureRunAsConnection	AzureServicePrin

To the right, a 'New Connection' dialog is open. It has fields for 'Name' (set to 'MyConnection'), 'Description' (empty), 'Type' (set to 'MyModuleConnection'), 'Username' (set to 'johndoe'), and 'Password' (represented by a masked input field). A 'Create' button is at the bottom right of the dialog.

The following runbook example uses a Contoso connection asset called `ContosoConnection` to access Contoso resources and return data from the external service. In this example, the fields are mapped to the `UserName` and `Password` properties of a `PSCredential` object and then passed to the cmdlet.

```
$contosoConnection = Get-AutomationConnection -Name 'ContosoConnection'

$cred = New-Object -TypeName System.Management.Automation.PSCredential -ArgumentList
$contosoConnection.UserName, $contosoConnection.Password
Connect-Contoso -Credential $cred
}
```

An easier and better way to approach this behavior is by directly passing the connection object to the cmdlet:

```
$contosoConnection = Get-AutomationConnection -Name 'ContosoConnection'

Connect-Contoso -Connection $contosoConnection
}
```

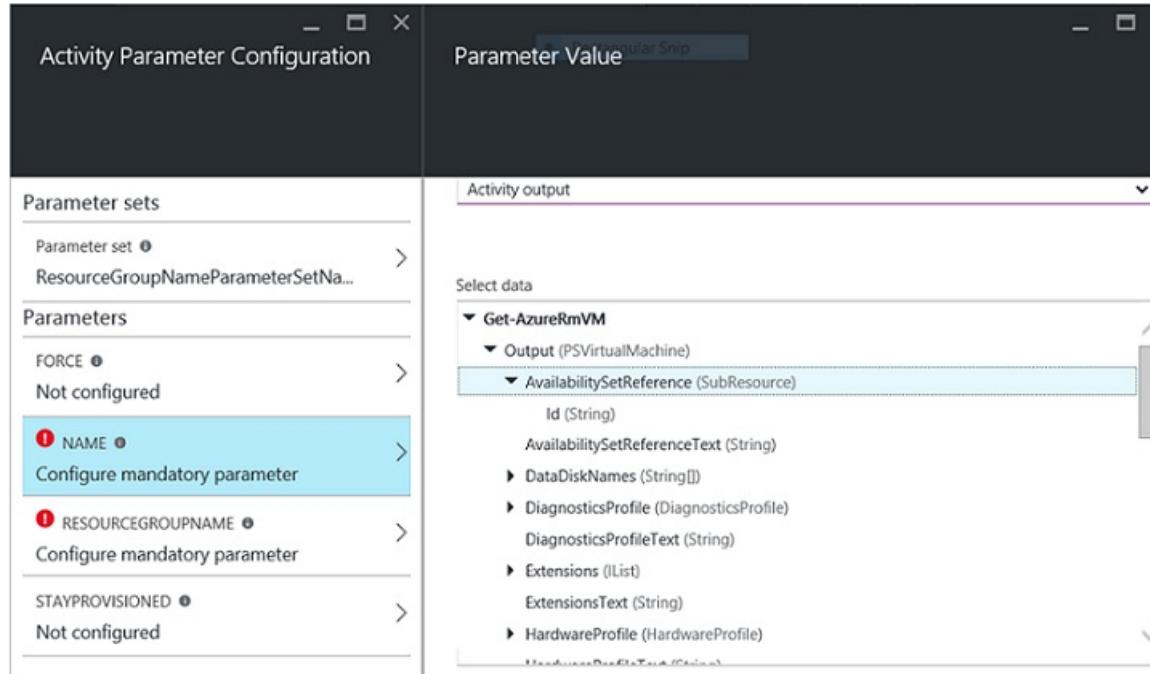
You can enable similar behavior for your cmdlets by allowing them to accept a connection object directly as a parameter, instead of just connection fields for parameters. Usually you want a parameter set for each, so that a user who isn't using Automation can call your cmdlets without constructing a hashtable to act as the connection object. The parameter set `UserAccount` is used to pass the connection field properties. `ConnectionObject` lets you pass the connection straight through.

Output type

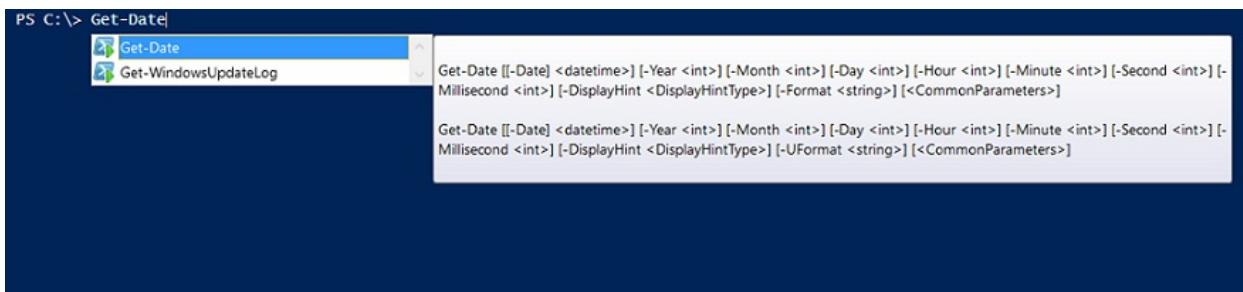
Define the output type for all cmdlets in your module. Defining an output type for a cmdlet allows design-time IntelliSense to help determine the output properties of the cmdlet during authoring. This practice is especially helpful during graphical runbook authoring, for which design-time knowledge is key to an easy user experience with your module.

Add `[OutputType([<MyOutputType>])]`, where `MyOutputType` is a valid type. To learn more about `OutputType`, see [About Functions OutputTypeAttribute](#). The following code is an example of adding `OutputType` to a cmdlet:

```
function Get-ContosoUser {
[OutputType([String])]
param(
    [string]
    $Parameter1
)
# <script location here>
}
```



This behavior is similar to the "type ahead" functionality of a cmdlet's output in the PowerShell integration service environment, without having to run it.



Cmdlet state

Make all cmdlets in your module stateless. Multiple runbook jobs can simultaneously run in the same `AppDomain` and the same process and sandbox. If there is any state shared on those levels, jobs can affect each other. This behavior can lead to intermittent and hard-to-diagnose issues. Here is an example of what not to do:

```
$globalNum = 0
function Set-GlobalNum {
    param(
        [int] $num
    )

    $globalNum = $num
}
function Get-GlobalNumTimesTwo {
    $output = $globalNum * 2

    $output
}
```

Module dependency

Ensure that the module is fully contained in a package that can be copied by using xcopy. Automation modules are distributed to the Automation sandboxes when runbooks execute. The modules must work independently of the host that runs them.

You should be able to zip up and move a module package, and have it function as normal when it's imported into another host's PowerShell environment. For this to happen, ensure that the module doesn't depend on any files outside the module folder that is zipped up when the module is imported into Automation.

Your module shouldn't depend on any unique registry settings on a host. Examples are the settings that are made when a product is installed.

Module file paths

Make sure that all files in the module have paths with fewer than 140 characters. Any paths over 140 characters in length cause problems with importing runbooks. Automation can't import a file with path size over 140 characters into the PowerShell session with `Import-Module`.

Import modules

This section defines several ways that you can import a module into your Automation account.

Import modules in the Azure portal

To import a module in the Azure portal:

1. In the portal, search for and select **Automation Accounts**.
2. On the **Automation Accounts** page, select your Automation account from the list.
3. Under **Shared Resources**, select **Modules**.
4. Select **Add a module**.

5. Select the .zip file that contains your module.

6. Select OK to start to import process.

Import modules by using PowerShell

You can use the [New-AzAutomationModule](#) cmdlet to import a module into your Automation account. The cmdlet takes a URL for a module .zip package.

```
New-AzAutomationModule -Name <ModuleName> -ContentLinkUri <ModuleUri> -ResourceGroupName <ResourceGroupName>  
-AutomationAccountName <AutomationAccountName>
```

You can also use the same cmdlet to import a module from the PowerShell Gallery directly. Make sure to grab `ModuleName` and `ModuleVersion` from the [PowerShell Gallery](#).

```
$moduleName = <ModuleName>  
$moduleVersion = <ModuleVersion>  
New-AzAutomationModule -AutomationAccountName <AutomationAccountName> -ResourceGroupName <ResourceGroupName>  
-Name $moduleName -ContentLinkUri  
"https://www.powershellgallery.com/api/v2/package/$moduleName/$moduleVersion"
```

Import modules from the PowerShell Gallery

You can import [PowerShell Gallery](#) modules either directly from the Gallery or from your Automation account.

To import a module directly from the PowerShell Gallery:

1. Go to <https://www.powershellgallery.com> and search for the module to import.
2. Under **Installation Options**, on the **Azure Automation** tab, select **Deploy to Azure Automation**. This action opens the Azure portal.
3. On the Import page, select your Automation account, and select OK.

The screenshot shows the PowerShell Gallery interface for the **Az.Automation** module. At the top, there's a navigation bar with links for **Packages**, **Publish**, **Statistics**, **Documentation**, and **Sign in**. Below the navigation is a search bar with the placeholder "Search PowerShell packages...". The main content area displays the module details for **Az.Automation** version 1.1.1. It includes a blue icon of a command prompt, download statistics (177,683), and a link to view full stats. The **Info** section lists project site, license info, contact owners, and a report link. The **Installation Options** section has tabs for **Install Module**, **Azure Automation** (which is selected and highlighted in grey), and **Manual Download**. A note states: "You can deploy this package directly to Azure Automation. Note that deploying packages with dependencies will delay all the dependencies to Azure Automation." Below this is a prominent blue button with white text that says **Deploy to Azure Automation**, which is also highlighted with a red box. Other sections visible include **Author(s)** (Microsoft Corporation), **Copyright** (Microsoft Corporation. All rights reserved.), and a link to **Package Details**.

To import a PowerShell Gallery module directly from your Automation account:

1. In the portal, search for and select **Automation Accounts**.
2. On the **Automation Accounts** page, select your Automation account from the list.
3. Under **Shared Resources**, select **Modules**.
4. Select **Browse gallery**, and then search the Gallery for a module.
5. Select the module to import, and select **Import**.
6. Select **OK** to start the import process.

The screenshot shows two windows side-by-side. The left window is titled 'Browse Gallery' and has a search bar containing 'azautomation'. Below the search bar is a list item for 'Az.Automation'. The right window is titled 'Az.Automation' and shows its details. At the top right of this window is a red box around the 'Import' button. Below the button, the module's description is visible, along with its version (1.1.1), download count (175,591), and last updated date (2/26/2019). A 'Content' section below lists several cmdlets:

TYPE	NAME
Cmdlet	Get-AzAutomationHybridWorkerGroup
Cmdlet	Remove-AzAutomationHybridWorkerGroup
Cmdlet	Get-AzAutomationJobOutputRecord
Cmdlet	Import-AzAutomationDscNodeConfiguration

Delete modules

If you have problems with a module, or you need to roll back to a previous version of a module, you can delete it from your Automation account. You can't delete the original versions of the **default modules** that are imported when you create an Automation account. If the module to delete is a newer version of one of the **default modules**, it rolls back to the version that was installed with your Automation account. Otherwise, any module you delete from your Automation account is removed.

Delete modules in the Azure portal

To remove a module in the Azure portal:

1. In the portal, search for and select **Automation Accounts**.
2. On the **Automation Accounts** page, select your Automation account from the list.
3. Under **Shared Resources**, select **Modules**.
4. Select the module you want to remove.
5. On the Module page, select **Delete**. If this module is one of the **default modules**, it rolls back to the version that existed when the Automation account was created.

Delete modules by using PowerShell

To remove a module through PowerShell, run the following command:

```
Remove-AzAutomationModule -Name <moduleName> -AutomationAccountName <automationAccountName> -  
ResourceGroupName <resourceGroupName>
```

Next steps

- For more information about using Azure PowerShell modules, see [Get started with Azure PowerShell](#).
- To learn more about creating PowerShell modules, see [Writing a Windows PowerShell module](#).

Update Azure PowerShell modules

4/22/2021 • 2 minutes to read • [Edit Online](#)

The most common PowerShell modules are provided by default in each Automation account. See [Default modules](#). As the Azure team updates the Azure modules regularly, changes can occur with the included cmdlets. These changes, for example, renaming a parameter or deprecating a cmdlet entirely, can negatively affect your runbooks.

NOTE

You can't delete global modules, which are modules that Automation provides out of the box.

Set up an Automation account

To avoid impacting your runbooks and the processes they automate, be sure to test and validate as you make updates. If you don't have a dedicated Automation account intended for this purpose, consider creating one so that you can test many different scenarios during the development of your runbooks. This testing should include iterative changes, such as updating the PowerShell modules.

Make sure that your Automation account has an [Azure Run As account](#) created.

If you develop your scripts locally, it's recommended to have the same module versions locally that you have in your Automation account when testing to ensure that you receive the same results. After the results are validated and you've applied any changes required, you can move the changes to production.

NOTE

A new Automation account might not contain the latest modules.

Obtain a runbook to use for updates

To update the Azure modules in your Automation account, you must use the [Update-AutomationAzureModulesForAccount](#) runbook, which is available as open source. To start using this runbook to update your Azure modules, download it from the GitHub repository. You can then import it into your Automation account or run it as a script. To learn how to import a runbook in your Automation account, see [Import a runbook](#).

The [Update-AutomationAzureModulesForAccount](#) runbook supports updating the Azure, Azurerm, and Az modules by default. Review the [Update Azure modules runbook README](#) for more information on updating Az.Automation modules with this runbook. There are additional important factors that you need to take into account when using the Az modules in your Automation account. To learn more, see [Manage modules in Azure Automation](#).

Use update runbook code as a regular PowerShell script

You can use the runbook code as a regular PowerShell script instead of a runbook. To do this, sign in to Azure using the [Connect-AzAccount](#) cmdlet first, then pass `-Login $false` to the script.

Use the update runbook on sovereign clouds

To use this runbook on sovereign clouds, use the `AzEnvironment` parameter to pass the correct environment to the runbook. Acceptable values are AzureCloud (Azure public cloud), AzureChinaCloud, AzureGermanCloud, and AzureUSGovernment. These values can be retrieved using `Get-AzEnvironment | select Name`. If you don't pass a value to this cmdlet, the runbook defaults to AzureCloud.

Use the update runbook to update a specific module version

If you want to use a specific Azure PowerShell module version instead of the latest module available on the PowerShell Gallery, pass these versions to the optional `ModuleVersionOverrides` parameter of the **Update-AutomationAzureModulesForAccount** runbook. For examples, see the [Update-AutomationAzureModulesForAccount.ps1](#) runbook. Azure PowerShell modules that aren't mentioned in the `ModuleVersionOverrides` parameter are updated with the latest module versions on the PowerShell Gallery. If you pass nothing to the `ModuleVersionOverrides` parameter, all modules are updated with the latest module versions on the PowerShell Gallery. This behavior is the same for the **Update Azure Modules** button in the Azure portal.

Next steps

- For details of using modules, see [Manage modules in Azure Automation](#).
- For information about the update runbook, see [Update Azure modules runbook](#).

Manage schedules in Azure Automation

4/22/2021 • 8 minutes to read • [Edit Online](#)

To schedule a runbook in Azure Automation to start at a specified time, you link it to one or more schedules. A schedule can be configured to either run once or on a recurring hourly or daily schedule for runbooks in the Azure portal. You can also schedule them for weekly, monthly, specific days of the week or days of the month, or a particular day of the month. A runbook can be linked to multiple schedules, and a schedule can have multiple runbooks linked to it.

NOTE

Azure Automation supports Daylight Savings Time and schedules it appropriately for automation operations.

NOTE

Schedules currently are not enabled for Azure Automation DSC configurations.

PowerShell cmdlets used to access schedules

The cmdlets in the following table create and manage Automation schedules with PowerShell. They ship as part of the [Az modules](#).

CMDLETS	DESCRIPTION
Get-AzAutomationSchedule	Retrieves a schedule.
Get-AzAutomationScheduledRunbook	Retrieves scheduled runbooks.
New-AzAutomationSchedule	Creates a new schedule.
Register-AzAutomationScheduledRunbook	Associates a runbook with a schedule.
Remove-AzAutomationSchedule	Removes a schedule.
Set-AzAutomationSchedule	Sets the properties for an existing schedule.
Unregister-AzAutomationScheduledRunbook	Dissociates a runbook from a schedule.

Create a schedule

You can create a new schedule for your runbooks from the Azure portal, with PowerShell, or using an Azure Resource Manager (ARM) template. To avoid affecting your runbooks and the processes they automate, you should first test any runbooks that have linked schedules with an Automation account dedicated for testing. A test validates that your scheduled runbooks continue to work correctly. If you see a problem, you can troubleshoot and apply any changes required before you migrate the updated runbook version to production.

NOTE

Your Automation account doesn't automatically get any new versions of modules unless you've updated them manually by selecting the [Update Azure modules](#) option from **Modules**. Azure Automation uses the latest modules in your Automation account when a new scheduled job is run.

Create a new schedule in the Azure portal

1. From your Automation account, on the left-hand pane select **Schedules** under **Shared Resources**.
2. On the **Schedules** page, select **Add a schedule**.
3. On the **New schedule** page, enter a name and optionally enter a description for the new schedule.

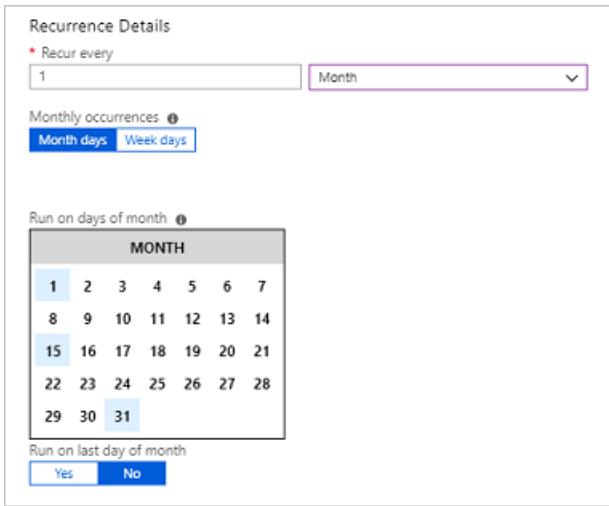
NOTE

Automation schedules do not currently support using special characters in the schedule name.

4. Select whether the schedule runs once or on a reoccurring schedule by selecting **Once** or **Recurring**. If you select **Once**, specify a start time and then select **Create**. If you select **Recurring**, specify a start time. For **Recur every**, select how often you want the runbook to repeat. Select by hour, day, week, or month.
 - If you select **Week**, the days of the week are presented for you to choose from. Select as many days as you want. The first run of your schedule will happen on the first day selected after the start time. For example, to choose a weekend schedule, select Saturday and Sunday.

The screenshot shows the 'Recur every' configuration for a schedule. The 'Once' tab is selected. In the 'Recur every' section, '1' is entered in the input field and 'Week' is selected in the dropdown. Below this, the 'On these days' section is expanded, showing checkboxes for each day of the week. 'Saturday' and 'Sunday' are checked, while the other days (Monday, Tuesday, Wednesday, Thursday, Friday) are unchecked.

- If you select **Month**, you're given different options. For the **Monthly occurrences** option, select either **Month days** or **Week days**. If you select **Month days**, a calendar appears so that you can choose as many days as you want. If you choose a date such as the 31st that doesn't occur in the current month, the schedule won't run. If you want the schedule to run on the last day, select **Yes** under **Run on last day of month**. If you select **Week days**, the **Recur every** option appears. Choose **First**, **Second**, **Third**, **Fourth**, or **Last**. Finally, choose a day to repeat on.



- When you're finished, select **Create**.

Create a new schedule with PowerShell

Use the [New-AzAutomationSchedule](#) cmdlet to create schedules. You specify the start time for the schedule and the frequency it should run. The following examples show how to create many different schedule scenarios.

NOTE

Automation schedules do not currently support using special characters in the schedule name.

Create a one-time schedule

The following example creates a one-time schedule.

```
$TimeZone = ([System.TimeZoneInfo]::Local).Id  
New-AzAutomationSchedule -AutomationAccountName "ContosoAutomation" -Name "Schedule01" -StartTime "23:00" -  
OneTime -ResourceGroupName "ResourceGroup01" -TimeZone $TimeZone
```

Create a recurring schedule

The following example shows how to create a recurring schedule that runs every day at 1:00 PM for a year.

```
$StartTime = Get-Date "13:00:00"  
$EndTime = $StartTime.AddYears(1)  
New-AzAutomationSchedule -AutomationAccountName "ContosoAutomation" -Name "Schedule02" -StartTime $StartTime  
-ExpiryTime $EndTime -DayInterval 1 -ResourceGroupName "ResourceGroup01"
```

Create a weekly recurring schedule

The following example shows how to create a weekly schedule that runs on weekdays only.

```
$StartTime = (Get-Date "13:00:00").AddDays(1)  
[System.DayOfWeek[]]$WeekDays = @([System.DayOfWeek]::Monday..[System.DayOfWeek]::Friday)  
New-AzAutomationSchedule -AutomationAccountName "ContosoAutomation" -Name "Schedule03" -StartTime $StartTime  
-WeekInterval 1 -DaysOfWeek $WeekDays -ResourceGroupName "ResourceGroup01"
```

Create a weekly recurring schedule for weekends

The following example shows how to create a weekly schedule that runs on weekends only.

```
$StartTime = (Get-Date "18:00:00").AddDays(1)  
[System.DayOfWeek[]]$WeekendDays = @([System.DayOfWeek]::Saturday,[System.DayOfWeek]::Sunday)  
New-AzAutomationSchedule -AutomationAccountName "ContosoAutomation" -Name "Weekends 6PM" -StartTime  
$StartTime -WeekInterval 1 -DaysOfWeek $WeekendDays -ResourceGroupName "ResourceGroup01"
```

Create a recurring schedule for the first, fifteenth, and last days of the month

The following example shows how to create a recurring schedule that runs on the first, fifteenth, and last day of a month.

```
$StartTime = (Get-Date "18:00:00").AddDays(1)
New-AzAutomationSchedule -AutomationAccountName "TestAzureAuto" -Name "1st, 15th and Last" -StartTime
$StartTime -DaysOfMonth @("One", "Fifteenth", "Last") -ResourceGroupName "TestAzureAuto" -MonthInterval 1
```

Create a schedule with a Resource Manager template

In this example, we use an Automation Resource Manager (ARM) template that creates a new job schedule. For general information about this template to manage Automation job schedules, see [Microsoft.Automation.automationAccounts/jobSchedules template reference](#).

Copy this template file into a text editor:

```
{
  "name": "5d5f3a05-111d-4892-8dcc-9064fa591b96",
  "type": "Microsoft.Automation/automationAccounts/jobSchedules",
  "apiVersion": "2015-10-31",
  "properties": {
    "schedule": {
      "name": "scheduleName"
    },
    "runbook": {
      "name": "runbookName"
    },
    "runOn": "hybridWorkerGroup",
    "parameters": {}
  }
}
```

Edit the following parameter values and save the template as a JSON file:

- Job schedule object name: A GUID (Globally Unique Identifier) is used as the name of the job schedule object.

IMPORTANT

For each job schedule deployed with an ARM template, the GUID must be unique. Even if you're rescheduling an existing schedule, you'll need to change the GUID. This applies even if you've previously deleted an existing job schedule that was created with the same template. Reusing the same GUID results in a failed deployment. There are services online that can generate a new GUID for you, such as this [Free Online GUID Generator](#).

- Schedule name: Represents the name of the Automation job schedule that will be linked to the specified runbook.
- Runbook name: Represents the name of the Automation runbook the job schedule is to be associated with.

Once the file has been saved, you can create the runbook job schedule with the following PowerShell command. The command uses the `TemplateFile` parameter to specify the path and filename of the template.

```
New-AzResourceGroupDeployment -ResourceGroupName "ContosoEngineering" -TemplateFile "
<path>\RunbookJobSchedule.json"
```

Link a schedule to a runbook

A runbook can be linked to multiple schedules, and a schedule can have multiple runbooks linked to it. If a runbook has parameters, you can provide values for them. You must provide values for any mandatory parameters, and you also can provide values for any optional parameters. These values are used each time the runbook is started by this schedule. You can attach the same runbook to another schedule and specify different parameter values.

Link a schedule to a runbook with the Azure portal

1. In the Azure portal, from your automation account, select **Runbooks** under **Process Automation**.
2. Select the name of the runbook to schedule.
3. If the runbook isn't currently linked to a schedule, you're offered the option to create a new schedule or link to an existing schedule.
4. If the runbook has parameters, you can select the option **Modify run settings (Default:Azure)** and the **Parameters** pane appears. You can enter parameter information here.

Link a schedule to a runbook with PowerShell

Use the [Register-AzAutomationScheduledRunbook](#) cmdlet to link a schedule. You can specify values for the runbook's parameters with the Parameters parameter. For more information on how to specify parameter values, see [Starting a Runbook in Azure Automation](#). The following example shows how to link a schedule to a runbook by using an Azure Resource Manager cmdlet with parameters.

```
$automationAccountName = "MyAutomationAccount"
$runbookName = "Test-Runbook"
$scheduleName = "Sample-DailySchedule"
$params = @{"FirstName"="Joe";"LastName"="Smith";"RepeatCount"=2;"Show"=$true}
Register-AzAutomationScheduledRunbook -AutomationAccountName $automationAccountName ` 
-Name $runbookName -ScheduleName $scheduleName -Parameters $params ` 
-ResourceGroupName "ResourceGroup01"
```

Schedule runbooks to run more frequently

The most frequent interval for which a schedule in Azure Automation can be configured is one hour. If you require schedules to run more frequently than that, there are two options:

- Create a [webhook](#) for the runbook, and use [Azure Logic Apps](#) to call the webhook. Azure Logic Apps provides more fine-grained granularity to define a schedule.
- Create four schedules that all start within 15 minutes of each other and run once every hour. This scenario allows the runbook to run every 15 minutes with the different schedules.

Disable a schedule

When you disable a schedule, any runbook linked to it no longer runs on that schedule. You can manually disable a schedule or set an expiration time for schedules with a frequency when you create them. When the expiration time is reached, the schedule is disabled.

Disable a schedule from the Azure portal

1. In your Automation account, on the left-hand pane select **Schedules** under **Shared Resources**.
2. Select the name of a schedule to open the details pane.
3. Change **Enabled** to **No**.

NOTE

If you want to disable a schedule that has a start time in the past, you must change the start date to a time in the future before you save it.

Disable a schedule with PowerShell

Use the [Set-AzAutomationSchedule](#) cmdlet to change the properties of an existing schedule. To disable the schedule, specify False for the `.IsEnabled` parameter.

The following example shows how to disable a schedule for a runbook by using an Azure Resource Manager cmdlet.

```
$automationAccountName = "MyAutomationAccount"
$scheduleName = "Sample-MonthlyDaysOfMonthSchedule"
Set-AzAutomationSchedule -AutomationAccountName $automationAccountName ` 
-Name $scheduleName -IsEnabled $false -ResourceGroupName "ResourceGroup01"
```

Remove a schedule

When you're ready to remove your schedules, you can either use the Azure portal or PowerShell. Remember that you can only remove a schedule that has been disabled as described in the previous section.

Remove a schedule using the Azure portal

1. In your Automation account, on the left-hand pane select **Schedules** under **Shared Resources**.
2. Select the name of a schedule to open the details pane.
3. Click **Delete**.

Remove a schedule with PowerShell

You can use the [Remove-AzAutomationSchedule](#) cmdlet as shown below to delete an existing schedule.

```
$automationAccountName = "MyAutomationAccount"
$scheduleName = "Sample-MonthlyDaysOfMonthSchedule"
Remove-AzAutomationSchedule -AutomationAccountName $automationAccountName ` 
-Name $scheduleName -ResourceGroupName "ResourceGroup01"
```

Next steps

- To learn more about the cmdlets used to access schedules, see [Manage modules in Azure Automation](#).
- For general information about runbooks, see [Runbook execution in Azure Automation](#).

Manage variables in Azure Automation

4/22/2021 • 7 minutes to read • [Edit Online](#)

Variable assets are values that are available to all runbooks and DSC configurations in your Automation account. You can manage them from the Azure portal, from PowerShell, within a runbook, or in a DSC configuration.

Automation variables are useful for the following scenarios:

- Sharing a value among multiple runbooks or DSC configurations.
- Sharing a value among multiple jobs from the same runbook or DSC configuration.
- Managing a value used by runbooks or DSC configurations from the portal or from the PowerShell command line. An example is a set of common configuration items, such as a specific list of VM names, a specific resource group, an AD domain name, and more.

Azure Automation persists variables and makes them available even if a runbook or DSC configuration fails. This behavior allows one runbook or DSC configuration to set a value that is then used by another runbook, or by the same runbook or DSC configuration the next time it runs.

Azure Automation stores each encrypted variable securely. When you create a variable, you can specify its encryption and storage by Azure Automation as a secure asset. After you create the variable, you can't change its encryption status without re-creating the variable. If you have Automation account variables storing sensitive data that are not already encrypted, then you need to delete them and recreate them as encrypted variables. An Azure Security Center recommendation is to encrypt all Azure Automation variables as described in [Automation account variables should be encrypted](#). If you have unencrypted variables that you want excluded from this security recommendation, see [Exempt a resource from recommendations and secure score](#) to create an exemption rule.

NOTE

Secure assets in Azure Automation include credentials, certificates, connections, and encrypted variables. These assets are encrypted and stored in Azure Automation using a unique key that is generated for each Automation account. Azure Automation stores the key in the system-managed Key Vault. Before storing a secure asset, Automation loads the key from Key Vault and then uses it to encrypt the asset.

Variable types

When you create a variable with the Azure portal, you must specify a data type from the dropdown list so that the portal can display the appropriate control for entering the variable value. The following are variable types available in Azure Automation:

- String
- Integer
- DateTime
- Boolean
- Null

The variable isn't restricted to the specified data type. You must set the variable using Windows PowerShell if you want to specify a value of a different type. If you indicate `Not defined`, the value of the variable is set to Null. You must set the value with the [Set-AzAutomationVariable](#) cmdlet or the internal `Set-AutomationVariable`

cmdlet. You use the `Set-AutomationVariable` in your runbooks that are intended to run in the Azure sandbox environment, or on a Windows Hybrid Runbook Worker.

You can't use the Azure portal to create or change the value for a complex variable type. However, you can provide a value of any type using Windows PowerShell. Complex types are retrieved as a [Newtonsoft.Json.LinqJProperty](#) for a Complex object type instead of a PSObject type [PSCustomObject](#).

You can store multiple values to a single variable by creating an array or hashtable and saving it to the variable.

NOTE

VM name variables can be a maximum of 80 characters. Resource group variables can be a maximum of 90 characters. See [Naming rules and restrictions for Azure resources](#).

PowerShell cmdlets to access variables

The cmdlets in the following table create and manage Automation variables with PowerShell. They ship as part of the [Az modules](#).

CMDLET	DESCRIPTION
Get-AzAutomationVariable	Retrieves the value of an existing variable. If the value is a simple type, that same type is retrieved. If it's a complex type, a <code>PSCustomObject</code> type is retrieved. ¹
New-AzAutomationVariable	Creates a new variable and sets its value.
Remove-AzAutomationVariable	Removes an existing variable.
Set-AzAutomationVariable	Sets the value for an existing variable.

¹ You can't use this cmdlet to retrieve the value of an encrypted variable. The only way to do this is by using the internal `Get-AutomationVariable` cmdlet in a runbook or DSC configuration. For example, to see the value of an encrypted variable, you might create a runbook to get the variable and then write it to the output stream:

```
$encryptvar = Get-AutomationVariable -Name TestVariable  
Write-output "The encrypted value of the variable is: $encryptvar"
```

Internal cmdlets to access variables

The internal cmdlets in the following table are used to access variables in your runbooks and DSC configurations. These cmdlets come with the global module `Orchestrator.AssetManagement.Cmdlets`. For more information, see [Internal cmdlets](#).

INTERNAL CMDLET	DESCRIPTION
<code>Get-AutomationVariable</code>	Retrieves the value of an existing variable.
<code>Set-AutomationVariable</code>	Sets the value for an existing variable.

NOTE

Avoid using variables in the `Name` parameter of `Get-AutomationVariable` cmdlet in a runbook or DSC configuration. Use of a variable can complicate the discovery of dependencies between runbooks and Automation variables at design time.

Python functions to access variables

The functions in the following table are used to access variables in a Python 2 and 3 runbook. Python 3 runbooks are currently in preview.

PYTHON FUNCTIONS	DESCRIPTION
<code>automationassets.get_automation_variable</code>	Retrieves the value of an existing variable.
<code>automationassets.set_automation_variable</code>	Sets the value for an existing variable.

NOTE

You must import the `automationassets` module at the top of your Python runbook to access the asset functions.

Create and get a variable

NOTE

If you want to remove the encryption for a variable, you must delete the variable and recreate it as unencrypted.

Create and get a variable using the Azure portal

1. From your Automation account, on the left-hand pane select **Variables** under **Shared Resources**.
2. On the **Variables** page, select **Add a variable**.
3. Complete the options on the **New Variable** page and then select **Create** to save the new variable.

NOTE

Once you have saved an encrypted variable, it can't be viewed in the portal. It can only be updated.

Create and get a variable in Windows PowerShell

Your runbook or DSC configuration uses the `New-AzAutomationVariable` cmdlet to create a new variable and set its initial value. If the variable is encrypted, the call should use the `Encrypted` parameter. Your script can retrieve the value of the variable using `Get-AzAutomationVariable`.

NOTE

A PowerShell script can't retrieve an encrypted value. The only way to do this is to use the internal `Get-AutomationVariable` cmdlet.

The following example shows how to create a string variable and then return its value.

```

$rgName = "ResourceGroup01"
$accountName = "MyAutomationAccount"
$variableValue = "My String"

New-AzAutomationVariable -ResourceGroupName "ResourceGroup01"
-AutomationAccountName "MyAutomationAccount" -Name 'MyStringVariable' ` 
-Encrypted $false -Value 'My String'
$string = (Get-AzAutomationVariable -ResourceGroupName "ResourceGroup01" ` 
-AutomationAccountName "MyAutomationAccount" -Name 'MyStringVariable').Value

```

The following example shows how to create a variable with a complex type and then retrieve its properties. In this case, a virtual machine object from [Get-AzVM](#) is used specifying a subset of its properties.

```

$rgName = "ResourceGroup01"
$accountName = "MyAutomationAccount"

$vml = Get-AzVM -ResourceGroupName "ResourceGroup01" -Name "VM01" | Select Name, Location, Extensions
New-AzAutomationVariable -ResourceGroupName "ResourceGroup01" -AutomationAccountName "MyAutomationAccount" - 
Name "MyComplexVariable" -Encrypted $false -Value $vm

$vmValue = Get-AzAutomationVariable -ResourceGroupName "ResourceGroup01" ` 
-AutomationAccountName "MyAutomationAccount" -Name "MyComplexVariable"

$vmName = $vmValue.Value.Name
$vmTags = $vmValue.Value.Tags

```

Textual runbook examples

- [PowerShell](#)
- [Python 2](#)
- [Python 3](#)

The following example shows how to set and retrieve a variable in a textual runbook. This example assumes the creation of integer variables named **numberOfIterations** and **numberOfRunnings** and a string variable named **sampleMessage**.

```

$rgName = "ResourceGroup01"
$accountName = "MyAutomationAccount"

$numberOfIterations = Get-AutomationVariable -Name "numberOfIterations"
$numberOfRunnings = Get-AutomationVariable -Name "numberOfRunnings"
$sampleMessage = Get-AutomationVariable -Name "sampleMessage"

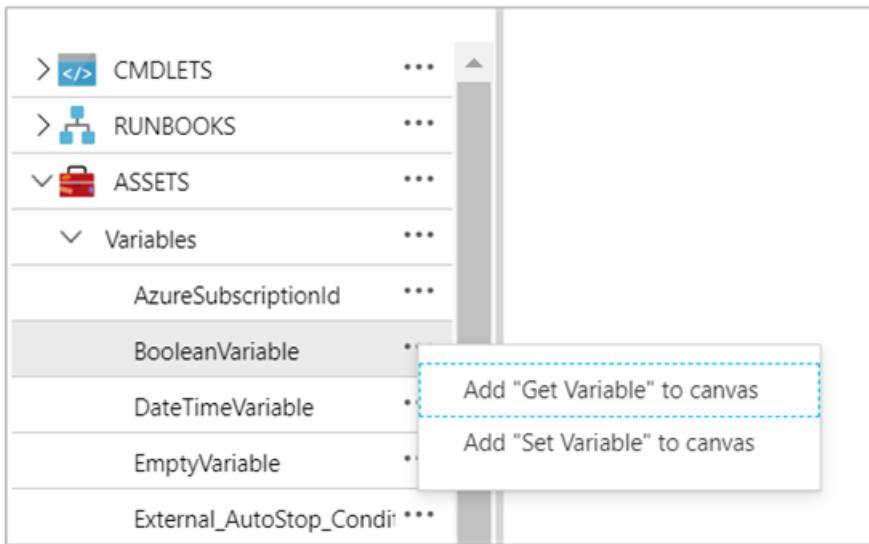
Write-Output "Runbook has been run $numberOfRunnings times."

for ($i = 1; $i -le $numberOfIterations; $i++) {
    Write-Output "$i: $sampleMessage"
}
Set-AutomationVariable -Name numberOfRunnings -Value ($numberOfRunnings += 1)

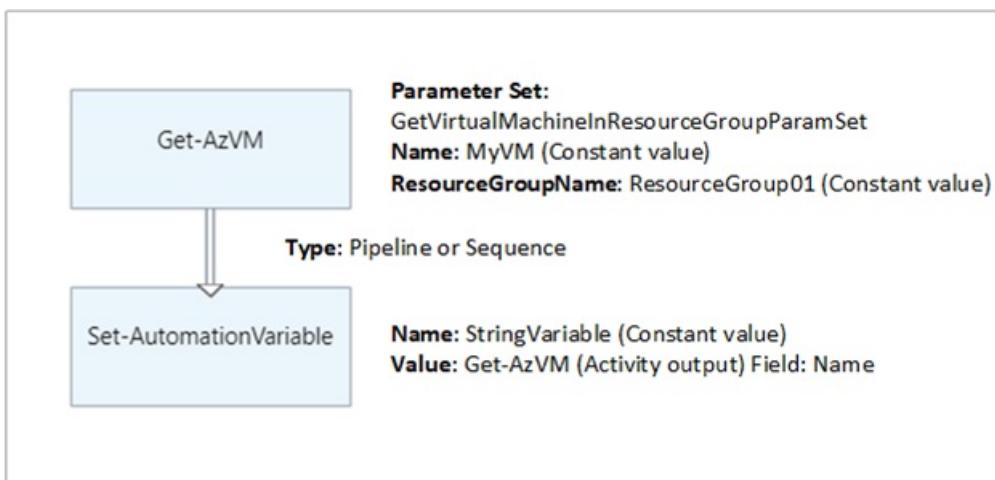
```

Graphical runbook examples

In a graphical runbook, you can add activities for the internal cmdlets **Get-AutomationVariable** or **Set-AutomationVariable**. Just right-click each variable in the Library pane of the graphical editor and select the activity that you want.



The following image shows example activities to update a variable with a simple value in a graphical runbook. In this example, the activity for `Get-AzVM` retrieves a single Azure virtual machine and saves the computer name to an existing Automation string variable. It doesn't matter whether the [link is a pipeline or sequence](#) since the code only expects a single object in the output.



Next steps

- To learn more about the cmdlets used to access variables, see [Manage modules in Azure Automation](#).
- For general information about runbooks, see [Runbook execution in Azure Automation](#).
- For details of DSC configurations, see [Azure Automation State Configuration overview](#).

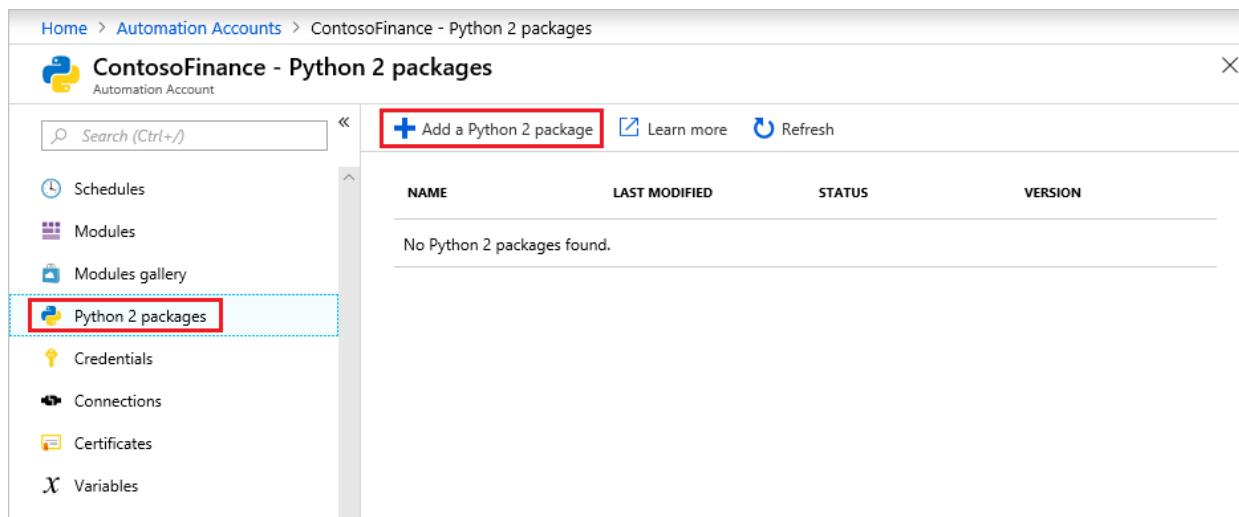
Manage Python 2 packages in Azure Automation

9/10/2021 • 2 minutes to read • [Edit Online](#)

Azure Automation allows you to run Python 2 runbooks on Azure and on Linux Hybrid Runbook Workers. To help in simplification of runbooks, you can use Python packages to import the modules that you need. This article describes how to manage and use Python packages in Azure Automation.

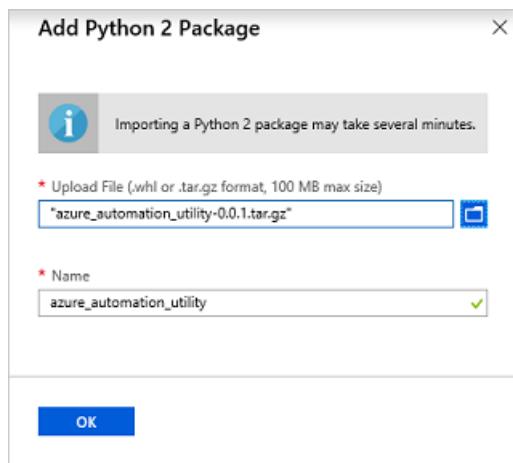
Import packages

In your Automation account, select **Python 2 packages** under Shared Resources. Click **+ Add a Python 2 package**.



The screenshot shows the 'ContosoFinance - Python 2 packages' page in the Azure portal. On the left, there's a sidebar with various options like Schedules, Modules, and Python 2 packages. The 'Python 2 packages' option is highlighted with a red box and has a blue dashed border around it. At the top right of the main content area, there's a red box highlighting the '+ Add a Python 2 package' button. Below it, there are 'Learn more' and 'Refresh' buttons. The main table has columns for NAME, LAST MODIFIED, STATUS, and VERSION. It displays the message 'No Python 2 packages found.'

On the Add Python 2 Package page, select a local package to upload. The package can be a .whl or .tar.gz file. When the package is selected, click **OK** to upload it.



The screenshot shows the 'Add Python 2 Package' dialog box. It includes an information icon with the text 'Importing a Python 2 package may take several minutes.' Below that is a file upload field with the path '* azure_automation_utility-0.0.1.tar.gz'. There's also a 'Name' input field where 'azure_automation_utility' is typed. At the bottom is a blue 'OK' button.

Once a package has been imported, it's listed on the Python 2 packages page in your Automation account. If you need to remove a package, select the package and click **Delete**.

The screenshot shows the Azure portal's Automation Accounts page for the 'ContosoFinance' account. Under the 'Python 2 packages' section, there are two entries:

NAME	LAST MODIFIED	STATUS	VERSION
azure_automation_utility	9/11/2018 10:38 AM	Available	0.0.1
ipaddress	9/11/2018 10:01 AM	Available	1.0.22

Import packages with dependencies

Azure automation doesn't resolve dependencies for Python packages during the import process. There are two ways to import a package with all its dependencies. Only one of the following steps needs to be used to import the packages into your Automation account.

Manually download

On a Windows 64-bit machine with [Python2.7](#) and [pip](#) installed, run the following command to download a package and all its dependencies:

```
C:\Python27\Scripts\pip2.7.exe download -d <output dir> <package name>
```

Once the packages are downloaded, you can import them into your automation account.

Runbook

To obtain a runbook, [import Python 2 packages from pypi into Azure Automation account](#) from the Azure Automation GitHub organization into your Automation account. Make sure the Run Settings are set to **Azure** and start the runbook with the parameters. The runbook requires a Run As account for the Automation account to work. For each parameter make sure you start it with the switch as seen in the following list and image:

- -s <subscriptionId>
- -g <resourceGroup>
- -a <automationAccount>
- -m <modulePackage>

The screenshot shows the Azure portal's Runbooks page for the 'TestAzureAuto' resource group. A runbook named 'import_py2package_from_pypi' is selected. The 'Start Runbook' dialog is open, showing the following parameters:

Parameter	Value
-s	00000000-0000-0000-0000-000000000000
-g	TestAzureAuto
-a	TestAzureAuto
-m	Azure

The 'Run Settings' section indicates the runbook will be executed on the 'Azure' hybrid worker.

The runbook allows you to specify what package to download. For example, use of the **Azure** parameter downloads all Azure modules and all dependencies (about 105).

Once the runbook is complete, you can check the [Python 2 packages](#) under **Shared Resources** in your Automation account to verify that the package has been imported correctly.

Use a package in a runbook

With a package imported, you can use it in a runbook. The following example uses the [Azure Automation utility package](#). This package makes it easier to use Python with Azure Automation. To use the package, follow the instructions in the GitHub repository and add it to the runbook. For example, you can use

```
from azure_automation_utility import get_automation_runas_credential
```

to import the function for retrieving the Run As account.

```
import azure.mgmt.resource
import automationassets
from azure_automation_utility import get_automation_runas_credential

# Authenticate to Azure using the Azure Automation RunAs service principal
runas_connection = automationassets.get_automation_connection("AzureRunAsConnection")
azure_credential = get_automation_runas_credential()

# Initialize the resource management client with the RunAs credential and subscription
resource_client = azure.mgmt.resource.ResourceManagementClient(
    azure_credential,
    str(runas_connection["SubscriptionId"]))

# Get list of resource groups and print them out
groups = resource_client.resource_groups.list()
for group in groups:
    print group.name
```

NOTE

The Python `automationassets` package is not available on pypi.org, so it's not available for import onto a Windows machine.

Develop and test runbooks offline

To develop and test your Python 2 runbooks offline, you can use the [Azure Automation Python emulated assets](#) module on GitHub. This module allows you to reference your shared resources such as credentials, variables, connections, and certificates.

Next steps

To prepare a Python runbook, see [Create a Python runbook](#).

Manage Python 3 packages (preview) in Azure Automation

8/26/2021 • 3 minutes to read • [Edit Online](#)

Azure Automation allows you to run Python 3 runbooks (preview) on Azure Sandbox environment and on Linux Hybrid Runbook Workers. To help in simplification of runbooks, you can use Python packages to import the modules that you need. To import a single package, see [Import a package](#). To import a package with multiple packages, see [Import a package with dependencies](#). This article describes how to manage and use Python 3 packages (preview) in Azure Automation.

Packages as source files

Azure Automation supports only a Python package that only contains Python code and doesn't include other language extensions or code in other languages. However, the Azure Sandbox environment might not have the required compilers for C/C++ binaries, so it's recommended to use [wheel files](#) instead. The [Python Package Index \(PyPI\)](#) is a repository of software for the Python programming language. When selecting a Python 3 package to import into your Automation account from PyPI, note the following filename parts:

FILENAME PART	DESCRIPTION
cp38	Automation supports Python 3.8.x for Cloud Jobs.
amd64	Azure sandbox processes are Windows 64-bit architecture.

For example, if you wanted to import pandas, you could select a wheel file with a name similar as

`pandas-1.2.3-cp38-win_amd64.whl`.

Some Python packages available on PyPI don't provide a wheel file. In this case, download the source (.zip or .tar.gz file) and generate the wheel file using `pip`. For example, perform the following steps using a 64-bit machine with Python 3.8.x and wheel package installed:

1. Download the source file `pandas-1.2.4.tar.gz`.
2. Run pip to get the wheel file with the following command: `pip wheel --no-deps pandas-1.2.4.tar.gz`.

Import a package

In your Automation account, select **Python packages** under **Shared Resources**. Then select **+ Add a Python package**.

The screenshot shows the Azure Automation Python packages management interface. The left sidebar has a tree view with 'Runbooks gallery', 'Hybrid worker groups', 'Watcher tasks', 'Shared Resources' (selected), 'Schedules', 'Modules', 'Modules gallery', 'Python packages' (selected), and 'Credentials'. The main area has a search bar and buttons for '+ Add a Python package', 'Learn more', and 'Refresh'. It shows two tabs: 'Python 2 packages' (selected) and 'Python 3 packages (preview)'. A table lists packages with columns: Name, Last modified, Status, and Version. The message 'No Python 2 packages found.' is displayed.

On the Add Python Package page, select **Python 3** for the **Version**, and select a local package to upload. The package can be a .whl or .tar.gz file. When the package is selected, select **OK** to upload it.

Add Python Package

Importing a Python package may take several minutes.

Upload File (.whl or .tar.gz format, 100 MB max size) *

Name *

Python version *

OK

Once a package has been imported, it's listed on the Python packages page in your Automation account, under the **Python 3 packages (preview)** tab. If you need to remove a package, select the package and select **Delete**.

Dashboard > MAIC-AA-Pri

MAIC-AA-Pri | Python packages

Automation Account

Runbooks gallery Hybrid worker groups Watcher tasks

Schedules Modules Modules gallery Python packages Credentials

+ Add a Python package Learn more Refresh

Python 2 packages Python 3 packages (preview)

Name	Last modified	Status	Version
azure_python3	12/13/2020 6:10 PM	Available	0.11.1

Import a package with dependencies

You can import a Python 3 package and its dependencies by importing the following Python script into a Python 3 runbook, and then running it.

```
https://github.com/azureautomation/runbooks/blob/master/Utility/Python/import_py3package_from_pypi.py
```

Importing the script into a runbook

For information on importing the runbook, see [Import a runbook from the Azure portal](#). Copy the file from GitHub to storage that the portal can access before you run the import.

The **Import a runbook** page defaults the runbook name to match the name of the script. If you have access to

the field, you can change the name. Runbook type may default to Python 2. If it does, make sure to change it to Python 3.

 Import a runbook X

Runbook file * ⓘ
"import_py3package_from_pypi.py" 

Name * ⓘ
import_py3package_from_pypi 

Runbook type * ⓘ
Python 3 

Description

Executing the runbook to import the package and dependencies

After creating and publishing the runbook, run it to import the package. See [Start a runbook in Azure Automation](#) for details on executing the runbook.

The script (`import_py3package_from_pypi.py`) requires the following parameters.

PARAMETER	DESCRIPTION
subscription_id	Subscription ID of the Automation account
resource_group	Name of the resource group that the Automation account is defined in
automation_account	Automation account name
module_name	Name of the module to import from <code>pypi.org</code>

For more information on using parameters with runbooks, see [Work with runbook parameters](#).

Use a package in a runbook

With the package imported, you can use it in a runbook. Add the following code to list all the resource groups in an Azure subscription.

```

import os
import azure.mgmt.resource
import automationassets

def get_automation_runas_credential(runas_connection):
    from OpenSSL import crypto
    import binascii
    from msrestazure import azure_active_directory
    import adal

    # Get the Azure Automation RunAs service principal certificate
    cert = automationassets.get_automation_certificate("AzureRunAsCertificate")
    pks12_cert = crypto.load_pkcs12(cert)
    pem_pkey = crypto.dump_privatekey(crypto.FILETYPE_PEM,pks12_cert.get_privatekey())

    # Get run as connection information for the Azure Automation service principal
    application_id = runas_connection["ApplicationId"]
    thumbprint = runas_connection["CertificateThumbprint"]
    tenant_id = runas_connection["TenantId"]

    # Authenticate with service principal certificate
    resource ="https://management.core.windows.net/"
    authority_url = ("https://login.microsoftonline.com/" +tenant_id)
    context = adal.AuthenticationContext(authority_url)
    return azure_active_directory.AdalAuthentication(
        lambda: context.acquire_token_with_client_certificate(
            resource,
            application_id,
            pem_pkey,
            thumbprint)
    )

# Authenticate to Azure using the Azure Automation RunAs service principal
runas_connection = automationassets.get_automation_connection("AzureRunAsConnection")
azure_credential = get_automation_runas_credential(runas_connection)

# Initialize the resource management client with the RunAs credential and subscription
resource_client = azure.mgmt.resource.ResourceManagementClient(
    azure_credential,
    str(runas_connection["SubscriptionId"]))

# Get list of resource groups and print them out
groups = resource_client.resource_groups.list()
for group in groups:
    print(group.name)

```

NOTE

The Python `automationassets` package is not available on pypi.org, so it's not available for import onto a Windows machine.

Next steps

To prepare a Python runbook, see [Create a Python runbook](#).

Troubleshoot shared resource issues

4/28/2021 • 4 minutes to read • [Edit Online](#)

This article discusses issues that might arise when you're using [shared resources](#) in Azure Automation.

Modules

Scenario: A module is stuck during import

Issue

A module is stuck in the *Importing* state when you're importing or updating your Azure Automation modules.

Cause

Because importing PowerShell modules is a complex, multistep process, a module might not import correctly, and can be stuck in a transient state. To learn more about the import process, see [Importing a PowerShell module](#).

Resolution

To resolve this issue, you must remove the module that is stuck by using the [Remove-AzAutomationModule](#) cmdlet. You can then retry importing the module.

```
Remove-AzAutomationModule -Name ModuleName -ResourceGroupName ExampleResourceGroup -AutomationAccountName ExampleAutomationAccount -Force
```

Scenario: AzureRM modules are stuck during import after an update attempt

Issue

A banner with the following message stays in your account after trying to update your AzureRM modules:

```
Azure modules are being updated
```

Cause

There is a known issue with updating the AzureRM modules in an Automation account. Specifically, the problem occurs if the modules are in a resource group with a numeric name starting with 0.

Resolution

To update your AzureRM modules in your Automation account, the account must be in a resource group with an alphanumeric name. Resource groups with numeric names starting with 0 are unable to update AzureRM modules at this time.

Scenario: Module fails to import or cmdlets can't be executed after importing

Issue

A module fails to import, or it imports successfully, but no cmdlets are extracted.

Cause

Some common reasons that a module might not successfully import to Azure Automation are:

- The structure doesn't match the structure that Automation needs.
- The module depends on another module that hasn't been deployed to your Automation account.
- The module is missing its dependencies in the folder.
- The [New-AzAutomationModule](#) cmdlet is being used to upload the module, and you haven't provided the full storage path or haven't loaded the module by using a publicly accessible URL.

Resolution

Use any of these solutions to fix the issue:

- Make sure that the module follows the format: ModuleName.zip -> ModuleName or Version Number -> (ModuleName.psm1, ModuleName.psd1).
- Open the .psd1 file and see if the module has any dependencies. If it does, upload these modules to the Automation account.
- Make sure that any referenced .dll files are present in the module folder.

Scenario: Update-AzureModule.ps1 suspends when updating modules

Issue

When you're using the [Update-AzureModule.ps1](#) runbook to update your Azure modules, the module update process is suspended.

Cause

For this runbook, the default setting to determine how many modules are updated simultaneously is 10. The update process is prone to errors when too many modules are being updated at the same time.

Resolution

It's not common that all the AzureRM or Az modules are required in the same Automation account. You should only import the specific modules that you need.

NOTE

Avoid importing the entire `Az.Automation` or `AzureRM.Automation` module, which imports all contained modules.

If the update process suspends, add the `SimultaneousModuleImportJobCount` parameter to the [Update-AzureModules.ps1](#) script, and provide a lower value than the default of 10. If you implement this logic, try starting with a value of 3 or 5. `SimultaneousModuleImportJobCount` is a parameter of the [Update-AutomationAzureModulesForAccount](#) system runbook that is used to update Azure modules. If you make this adjustment, the update process runs longer, but has a better chance of completing. The following example shows the parameter and where to put it in the runbook:

```
$Body = @"
{
    "properties":{
        "runbook":{
            "name":"Update-AutomationAzureModulesForAccount"
        },
        "parameters":{
            ...
            "SimultaneousModuleImportJobCount":"3",
            ...
        }
    }
"}@
```

Run As accounts

Scenario: You're unable to create or update a Run As account

Issue

When you try to create or update a Run As account, you receive an error similar to the following:

```
You do not have permissions to create...
```

Cause

You don't have the permissions that you need to create or update the Run As account, or the resource is locked at a resource group level.

Resolution

To create or update a Run As account, you must have appropriate [permissions](#) to the various resources used by the Run As account.

If the problem is because of a lock, verify that the lock can be removed. Then go to the resource that is locked in Azure portal, right-click the lock, and select **Delete**.

Scenario: You receive the error "Unable to find an entry point named 'GetPerAdapterInfo' in DLL 'iplpapi.dll'" when executing a runbook

Issue

When you're executing a runbook, you receive the following exception:

```
Unable to find an entry point named 'GetPerAdapterInfo' in DLL 'iplpapi.dll'
```

Cause

This error is most likely caused by an incorrectly configured [Run As account](#).

Resolution

Make sure that your Run As account is properly configured. Then verify that you have the proper code in your runbook to authenticate with Azure. The following example shows a snippet of code to authenticate to Azure in a runbook by using a Run As account.

```
$connection = Get-AutomationConnection -Name AzureRunAsConnection
Connect-AzAccount -ServicePrincipal -Tenant $connection.TenantID ` 
-ApplicationID $connection.ApplicationID -CertificateThumbprint $connection.CertificateThumbprint
```

Next steps

If this article doesn't resolve your issue, try one of the following channels for additional support:

- Get answers from Azure experts through [Azure Forums](#).
- Connect with [@AzureSupport](#). This is the official Microsoft Azure account for connecting the Azure community to the right resources: answers, support, and experts.
- File an Azure support incident. Go to the [Azure support site](#), and select **Get Support**.

Use existing runbooks and modules

9/10/2021 • 5 minutes to read • [Edit Online](#)

Rather than creating your own runbooks and modules in Azure Automation, you can access scenarios that have already been built by Microsoft and the community. You can get Azure-related PowerShell and Python runbooks from the Runbook Gallery in the Azure portal, and [modules](#) and [runbooks](#) (which may or may not be specific to Azure) from the PowerShell Gallery. You can also contribute to the community by sharing [scenarios that you develop](#).

NOTE

The TechNet Script Center is retiring. All of the runbooks from Script Center in the Runbook gallery have been moved to our [Automation GitHub organization](#). For more information, see [Azure Automation Runbooks moving to GitHub](#).

Import runbooks from GitHub with the Azure portal

1. In the Azure portal, open your Automation account.
2. Select **Runbooks gallery** under **Process Automation**.
3. Select **Source: GitHub**.
4. You can use the filters above the list to narrow the display by publisher, type, and sort. Locate the gallery item you want and select it to view its details.

Runbook Name	Description	Created by	Ratings	Downloads	Last Updated
Start Azure V2 VMs	Graphical PowerShell runbook connects to Azure using an Automation Run As account and starts all V2 VMs in an Azure subscription or in a resource group or a single named V2 VM. You can attach a recurring schedule to this runbook to Tags: Azure Virtual Machines, Start VM, GraphicalPS	Azure Automation Product	4.5 of 5	160,206	10/23/2016
Stop-Start-AzureVM (Scheduled VM Shutdown/Startup)	PowerShell Workflow runbook	Pradeeban Raja	5 of 5	72,400	8/21/2018
Troubleshooting utility for Azure Automation Update Management Agent	Python 2 Runbook	Azure Automation Product	4 of 5	3,423	9/2/2020

5. To import an item, click **Import** on the details page.

Home > MAIC-AA-Pri >



Stop-Start-AzureVM (Scheduled VM Shutdown/Startup)

[View Source](#)

...

Import

This PowerShell Workflow runbook connects to Azure using an Automation Credential and Starts/Stops a VM/a list of VMs/All VMs in a Subscription in-parallel.

Created by: Pradeban Raja - **Microsoft**

Ratings: 5 of 5

Tags: Windows Azure Virtual Machines, Azure Automation, azure vm

72,400 downloads

[View Source Project](#)

Last updated: 8/21/2018

```
1 Workflow Stop-Start-AzureVM
2 {
3     Param
4     (
5         [Parameter(Mandatory=$true)][ValidateNotNullOrEmpty()]
```

6. Optionally, change the name of the runbook on the import blade, and then click OK to import the runbook.

[Home](#) > [MAIC-AA-Pri](#) > [Stop-Start-AzureVM \(Scheduled VM Shutdown/Startup\)](#) >

Import ...

Name * ⓘ

Runbook type ⓘ

PowerShell Workflow

Description

This PowerShell Workflow runbook
connects to Azure using an Automation
Credential and Starts/Stops a VM/a list of

OK

7. The runbook appears on the Runbooks tab for the Automation account.

Runbooks in the PowerShell Gallery

IMPORTANT

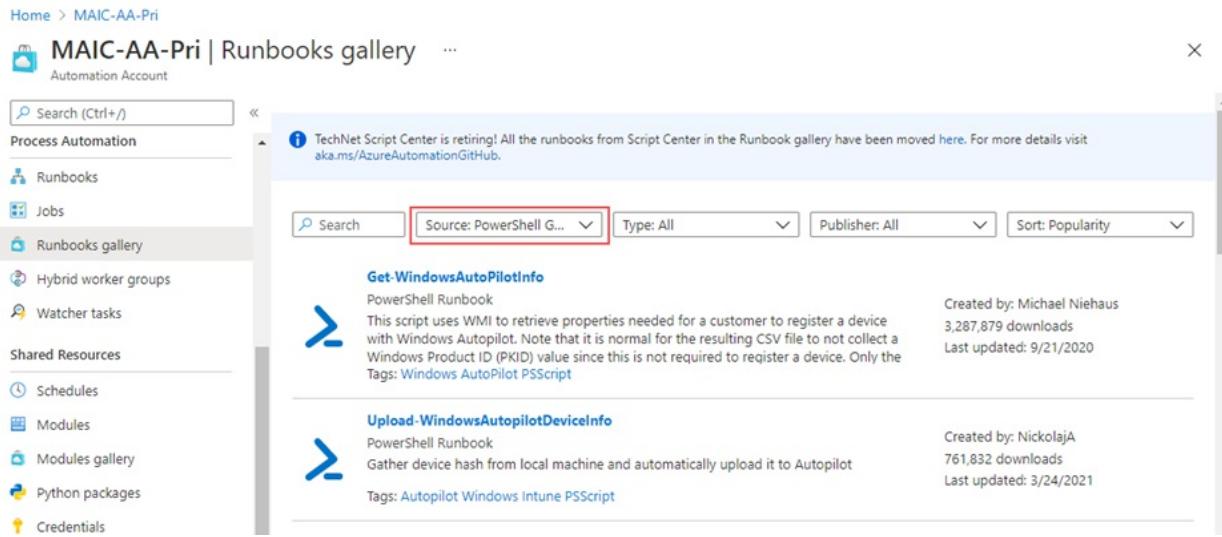
You should validate the contents of any runbooks that you get from the PowerShell Gallery. Use extreme caution in installing and running them in a production environment.

The [PowerShell Gallery](#) provides various runbooks from Microsoft and the community that you can import into Azure Automation. To use one, download a runbook from the gallery, or you can directly import runbooks from the gallery, or from your Automation account in the Azure portal.

NOTE

Graphical runbooks are not supported in PowerShell Gallery.

You can only import directly from the PowerShell Gallery using the Azure portal. You cannot perform this function using PowerShell. The procedure is the same as shown in [Import runbooks from GitHub with the Azure portal](#), except that the **Source** will be **PowerShell Gallery**.



Modules in the PowerShell Gallery

PowerShell modules contain cmdlets that you can use in your runbooks. Existing modules that you can install in Azure Automation are available in the [PowerShell Gallery](#). You can launch this gallery from the Azure portal and install the modules directly into Azure Automation, or you can manually download and install them.

You can also find modules to import in the Azure portal. They're listed for your Automation Account in the **Modules gallery** under **Shared resources**.

IMPORTANT

Do not include the keyword "AzureRm" in any script designed to be executed with the Az module. Inclusion of the keyword, even in a comment, may cause the AzureRm to load and then conflict with the Az module.

Common scenarios available in the PowerShell Gallery

The list below contains a few runbooks that support common scenarios. For a full list of runbooks created by the Azure Automation team, see [AzureAutomationTeam profile](#).

- [Update-ModulesInAutomationToLatestVersion](#) - Imports the latest version of all modules in an Automation account from PowerShell Gallery.
- [Enable-AzureDiagnostics](#) - Configures Azure Diagnostics and Log Analytics to receive Azure Automation logs containing job status and job streams.
- [Copy-ItemFromAzureVM](#) - Copies a remote file from a Windows Azure virtual machine.
- [Copy-ItemToAzureVM](#) - Copies a local file to an Azure virtual machine.

Contribute to the community

We strongly encourage you to contribute and help grow the Azure Automation community. Share the amazing runbooks you've built with the community. Your contributions will be appreciated!

Add a runbook to the GitHub Runbook gallery

You can add new PowerShell or Python runbooks to the Runbook gallery with this GitHub workflow.

1. Create a public repository on GitHub, and add the runbook and any other necessary files (like `readme.md`, `description`, and so on).
2. Add the topic `azureautomationrunbookgallery` to make sure the repository is discovered by our service, so it can be displayed in the Automation Runbook gallery.
3. If the runbook that you created is a PowerShell workflow, add the topic `PowerShellWorkflow`. If it's a Python 3 runbook, add `Python3`. No other specific topics are required for Azure runbooks, but we encourage you to add other topics that can be used for categorization and search in the Runbook Gallery.

NOTE

Check out existing runbooks in the gallery for things like formatting, headers, and existing tags that you might use (like `Azure Automation` or `Linux Azure Virtual Machines`).

To suggest changes to an existing runbook, file a pull request against it.

If you decide to clone and edit an existing runbook, best practice is to give it a different name. If you re-use the old name, it will show up twice in the Runbook gallery listing.

NOTE

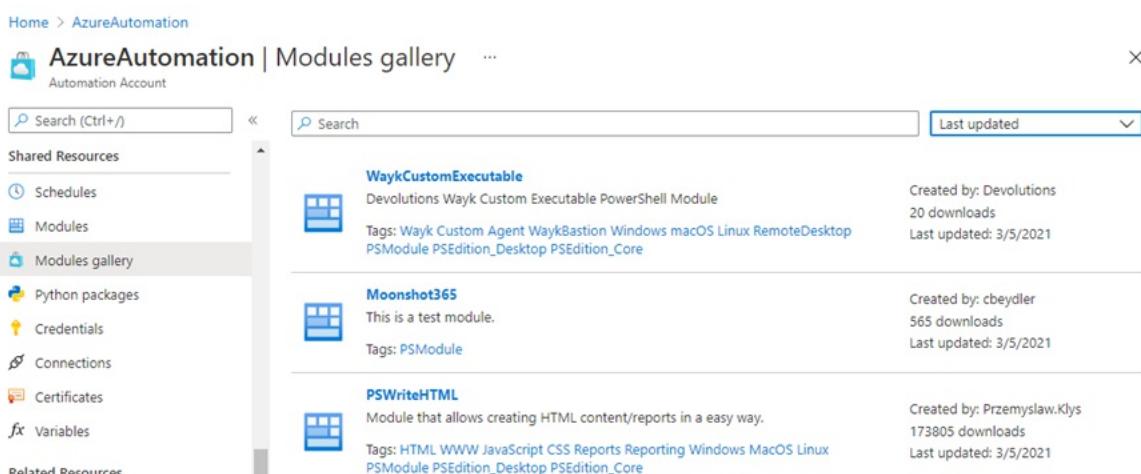
Please allow at least 12 hours for synchronization between GitHub and the Automation Runbook Gallery, for both updated and new runbooks.

Add a PowerShell runbook to the PowerShell gallery

Microsoft encourages you to add runbooks to the PowerShell Gallery that you think would be useful to other customers. The PowerShell Gallery accepts PowerShell modules and PowerShell scripts. You can add a runbook by [uploading it to the PowerShell Gallery](#).

Import a module from the Modules gallery in the Azure portal

1. In the Azure portal, open your Automation account.
2. Under **Shared Resources**, select **Modules gallery** to open the list of modules.



3. On the Browse gallery page, you can search by the following fields:

- Module Name

- Tags
 - Author
 - Cmdlet/DSC resource name
4. Locate a module that you're interested in and select it to view its details.

When you drill into a specific module, you can view more information. This information includes a link back to the PowerShell Gallery, any required dependencies, and all of the cmdlets or DSC resources that the module contains.

The screenshot shows the Azure Automation Import pane for the 'Posh-SSH' module. At the top, there's a breadcrumb navigation: Home > AzureAutomation > Browse Gallery >. Below that, the module name 'Posh-SSH' is displayed with a '...' button and a 'PowerShell Module' label. A prominent red-bordered 'Import' button is visible. A sub-section titled 'Provide SSH and SCP functionality for executing commands against remote hosts.' follows. On the left, there's a 'Learn more' section with links to the PowerShell Gallery and licensing information. On the right, module metadata is shown: Version 2.3.0, 4,125,729 downloads, and last updated on 10/4/2020. Below this, a 'Content' section lists cmdlets: Get-SCPFile, Get-SCPFolder, and Get-SCPItem.

5. To install the module directly into Azure Automation, click **Import**.
6. On the Import pane, you can see the name of the module to import. If all the dependencies are installed, the **OK** button is activated. If you're missing dependencies, you need to import those dependencies before you can import this module.
7. On the Import pane, click **OK** to import the module. While Azure Automation imports a module to your account, it extracts metadata about the module and the cmdlets. This action might take a couple of minutes since each activity needs to be extracted.
8. You receive an initial notification that the module is being deployed and another notification when it has completed.
9. After the module is imported, you can see the available activities. You can use module resources in your runbooks and DSC resources.

NOTE

Modules that only support PowerShell core are not supported in Azure Automation and are unable to be imported in the Azure portal, or deployed directly from the PowerShell Gallery.

Request a runbook or module

You can send requests to [User Voice](#). If you need help with writing a runbook or have a question about PowerShell, post a question to our [Microsoft Q&A question page](#).

Next steps

- To get started with PowerShell runbooks, see [Tutorial: Create a PowerShell runbook](#).

- To work with runbooks, see [Manage runbooks in Azure Automation](#).
- For more info on PowerShell scripting, see [PowerShell Docs](#).
- For a PowerShell cmdlet reference, see [Az.Automation](#).

Learn PowerShell Workflow for Azure Automation

4/22/2021 • 10 minutes to read • [Edit Online](#)

Runbooks in Azure Automation are implemented as Windows PowerShell workflows, Windows PowerShell scripts that use Windows Workflow Foundation. A workflow is a sequence of programmed, connected steps that perform long-running tasks or require the coordination of multiple steps across multiple devices or managed nodes.

While a workflow is written with Windows PowerShell syntax and launched by Windows PowerShell, it is processed by Windows Workflow Foundation. The benefits of a workflow over a normal script include simultaneous performance of an action against multiple devices and automatic recovery from failures.

NOTE

A PowerShell Workflow script is very similar to a Windows PowerShell script but has some significant differences that can be confusing to a new user. Therefore, we recommend that you write your runbooks using PowerShell Workflow only if you need to use [checkpoints](#).

For complete details of the topics in this article, see [Getting Started with Windows PowerShell Workflow](#).

Use Workflow keyword

The first step to converting a PowerShell script to a PowerShell workflow is enclosing it with the `Workflow` keyword. A workflow starts with the `Workflow` keyword followed by the body of the script enclosed in braces. The name of the workflow follows the `Workflow` keyword as shown in the following syntax:

```
Workflow Test-Workflow
{
    <Commands>
}
```

The name of the workflow must match the name of the Automation runbook. If the runbook is being imported, then the file name must match the workflow name and must end in `.ps1`.

To add parameters to the workflow, use the `Param` keyword just as you would in a script.

Learn differences between PowerShell Workflow code and PowerShell script code

PowerShell Workflow code looks almost identical to PowerShell script code except for a few significant changes. The following sections describe changes that you need to make to a PowerShell script for it to run in a workflow.

Activities

An activity is a specific task in a workflow that is performed in a sequence. Windows PowerShell Workflow automatically converts many of the Windows PowerShell cmdlets to activities when it runs a workflow. When you specify one of these cmdlets in your runbook, the corresponding activity is run by Windows Workflow Foundation.

If a cmdlet has no corresponding activity, Windows PowerShell Workflow automatically runs the cmdlet in an [InlineScript](#) activity. Some cmdlets are excluded and can't be used in a workflow unless you explicitly include

them in an [InlineScript](#) block. For more information, see [Using Activities in Script Workflows](#).

Workflow activities share a set of common parameters to configure their operation. See [about_WorkflowCommonParameters](#).

Positional parameters

You can't use positional parameters with activities and cmdlets in a workflow. Therefore, you must use parameter names. Consider the following code that gets all running services:

```
Get-Service | Where-Object {$_.Status -eq "Running"}
```

If you try to run this code in a workflow, you receive a message like

`Parameter set cannot be resolved using the specified named parameters.` To correct for this issue, provide the parameter name, as in the following example:

```
Workflow Get-RunningServices
{
    Get-Service | Where-Object -FilterScript {$_.Status -eq "Running"}
}
```

Deserialized objects

Objects in workflows are deserialized, meaning that their properties are still available, but not their methods. For example, consider the following PowerShell code, which stops a service using the `Stop` method of the `Service` object.

```
$Service = Get-Service -Name MyService
$Service.Stop()
```

If you try to run this in a workflow, you receive an error saying

`Method invocation is not supported in a Windows PowerShell Workflow.`

One option is to wrap these two lines of code in an [InlineScript](#) block. In this case, `Service` represents a service object within the block.

```
Workflow Stop-Service
{
    InlineScript {
        $Service = Get-Service -Name MyService
        $Service.Stop()
    }
}
```

Another option is to use another cmdlet that has the same functionality as the method, if one is available. In our example, the `Stop-Service` cmdlet provides the same functionality as the `Stop` method, and you might use the following code for a workflow.

```
Workflow Stop-MyService
{
    $Service = Get-Service -Name MyService
    Stop-Service -Name $Service.Name
}
```

Use [InlineScript](#)

The `InlineScript` activity is useful when you need to run one or more commands as traditional PowerShell script instead of PowerShell workflow. While commands in a workflow are sent to Windows Workflow Foundation for processing, commands in an `InlineScript` block are processed by Windows PowerShell.

`InlineScript` uses the following syntax shown below.

```
InlineScript
{
    <Script Block>
} <Common Parameters>
```

You can return output from an `InlineScript` by assigning the output to a variable. The following example stops a service and then outputs the service name.

```
Workflow Stop-MyService
{
    $Output = InlineScript {
        $Service = Get-Service -Name MyService
        $Service.Stop()
        $Service
    }

    $Output.Name
}
```

You can pass values into an `InlineScript` block, but you must use `$Using` scope modifier. The following example is identical to the previous example except that the service name is provided by a variable.

```
Workflow Stop-MyService
{
    $ServiceName = "MyService"

    $Output = InlineScript {
        $Service = Get-Service -Name $Using:ServiceName
        $Service.Stop()
        $Service
    }

    $Output.Name
}
```

While `InlineScript` activities might be critical in certain workflows, they do not support workflow constructs. You should use them only when necessary for the following reasons:

- You can't use [checkpoints](#) inside an `InlineScript` block. If a failure occurs within the block, it must resume from the beginning of the block.
- You can't use [parallel execution](#) inside an `InlineScript` block.
- `InlineScript` affects scalability of the workflow since it holds the Windows PowerShell session for the entire length of the `InlineScript` block.

For more information on using `InlineScript`, see [Running Windows PowerShell Commands in a Workflow](#) and [about_InlineScript](#).

Use parallel processing

One advantage of Windows PowerShell Workflows is the ability to perform a set of commands in parallel instead of sequentially as with a typical script.

You can use the `Parallel` keyword to create a script block with multiple commands that run concurrently. This uses the following syntax shown below. In this case, Activity1 and Activity2 starts at the same time. Activity3 starts only after both Activity1 and Activity2 have completed.

```
Parallel
{
    <Activity1>
    <Activity2>
}
<Activity3>
```

For example, consider the following PowerShell commands that copy multiple files to a network destination. These commands are run sequentially so that one file must finish copying before the next is started.

```
Copy-Item -Path C:\LocalPath\file1.txt -Destination \\NetworkPath\file1.txt
Copy-Item -Path C:\LocalPath\file2.txt -Destination \\NetworkPath\file2.txt
Copy-Item -Path C:\LocalPath\file3.txt -Destination \\NetworkPath\file3.txt
```

The following workflow runs these same commands in parallel so that they all start copying at the same time. Only after they are all copied is the completion message displayed.

```
Workflow Copy-Files
{
    Parallel
    {
        Copy-Item -Path "C:\LocalPath\file1.txt" -Destination "\\NetworkPath"
        Copy-Item -Path "C:\LocalPath\file2.txt" -Destination "\\NetworkPath"
        Copy-Item -Path "C:\LocalPath\file3.txt" -Destination "\\NetworkPath"
    }

    Write-Output "Files copied."
}
```

You can use the `ForEach -Parallel` construct to process commands for each item in a collection concurrently. The items in the collection are processed in parallel while the commands in the script block run sequentially. This process uses the following syntax shown below. In this case, Activity1 starts at the same time for all items in the collection. For each item, Activity2 starts after Activity1 is complete. Activity3 starts only after both Activity1 and Activity2 have completed for all items. We use the `ThrottleLimit` parameter to limit the parallelism. Too high of a `ThrottleLimit` can cause problems. The ideal value for the `ThrottleLimit` parameter depends on many factors in your environment. Start with a low value and try different increasing values until you find one that works for your specific circumstance.

```
ForEach -Parallel -ThrottleLimit 10 ($<item> in $<collection>)
{
    <Activity1>
    <Activity2>
}
<Activity3>
```

The following example is similar to the previous example copying files in parallel. In this case, a message is displayed for each file after it copies. Only after they are all copied is the final completion message displayed.

```

Workflow Copy-Files
{
    $files = @("C:\LocalPath\file1.txt","C:\LocalPath\file2.txt","C:\LocalPath\file3.txt")

    ForEach -Parallel -ThrottleLimit 10 ($File in $Files)
    {
        Copy-Item -Path $File -Destination \\NetworkPath
        Write-Output "$File copied."
    }

    Write-Output "All files copied."
}

```

NOTE

We do not recommend running child runbooks in parallel since this has been shown to give unreliable results. The output from the child runbook sometimes does not show up, and settings in one child runbook can affect the other parallel child runbooks. Variables such as `VerbosePreference`, `WarningPreference`, and others might not propagate to the child runbooks. And if the child runbook changes these values, they might not be properly restored after invocation.

Use checkpoints in a workflow

A checkpoint is a snapshot of the current state of the workflow that includes the current values for variables and any output generated to that point. If a workflow ends in error or is suspended, it starts from its last checkpoint the next time it runs, instead of starting at the beginning.

You can set a checkpoint in a workflow with the `Checkpoint-Workflow` activity. Azure Automation has a feature called [fair share](#), for which any runbook that runs for three hours is unloaded to allow other runbooks to run. Eventually, the unloaded runbook is reloaded. When it is, it resumes execution from the last checkpoint taken in the runbook.

To guarantee that the runbook eventually completes, you must add checkpoints at intervals that run for less than three hours. If during each run a new checkpoint is added, and if the runbook is evicted after three hours due to an error, the runbook is resumed indefinitely.

In the following example, an exception occurs after Activity2, causing the workflow to end. When the workflow is run again, it starts by running Activity2, since this activity was just after the last checkpoint set.

```

<Activity1>
Checkpoint-Workflow
<Activity2>
<Exception>
<Activity3>

```

Set checkpoints in a workflow after activities that might be prone to exception and should not be repeated if the workflow is resumed. For example, your workflow might create a virtual machine. You can set a checkpoint both before and after the commands to create the virtual machine. If the creation fails, then the commands are repeated if the workflow is started again. If the workflow fails after the creation succeeds, the virtual machine is not created again when the workflow is resumed.

The following example copies multiple files to a network location and sets a checkpoint after each file. If the network location is lost, then the workflow ends in error. When it is started again, it resumes at the last checkpoint. Only the files that have already been copied are skipped.

```

Workflow Copy-Files
{
    $files = @("C:\LocalPath\file1.txt","C:\LocalPath\file2.txt","C:\LocalPath\file3.txt")

    ForEach ($File in $Files)
    {
        Copy-Item -Path $File -Destination \\NetworkPath
        Write-Output "$File copied."
        Checkpoint-Workflow
    }

    Write-Output "All files copied."
}

```

Because user name credentials are not persisted after you call the [Suspend-Workflow](#) activity or after the last checkpoint, you need to set the credentials to null and then retrieve them again from the asset store after

[Suspend-Workflow](#) or checkpoint is called. Otherwise, you might receive the following error message:

The workflow job cannot be resumed, either because persistence data could not be saved completely, or saved persistence data has been corrupted. You must restart the workflow.

The following same code demonstrates how to handle this situation in your PowerShell Workflow runbooks.

```

workflow CreateTestVms
{
    $Cred = Get-AzAutomationCredential -Name "MyCredential"
    $null = Connect-AzAccount -Credential $Cred

    $VmsToCreate = Get-AzAutomationVariable -Name "VmsToCreate"

    foreach ($VmName in $VmsToCreate)
    {
        # Do work first to create the VM (code not shown)

        # Now add the VM
        New-AzVM -VM $Vm -Location "WestUS" -ResourceGroupName "ResourceGroup01"

        # Checkpoint so that VM creation is not repeated if workflow suspends
        $Cred = $null
        Checkpoint-Workflow
        $Cred = Get-AzAutomationCredential -Name "MyCredential"
        $null = Connect-AzAccount -Credential $Cred
    }
}

```

NOTE

For non-graphical PowerShell runbooks, [Add-AzAccount](#) and [Add-AzureRMAccount](#) are aliases for [Connect-AzAccount](#). You can use these cmdlets or you can [update your modules](#) in your Automation account to the latest versions. You might need to update your modules even if you have just created a new Automation account. Use of these cmdlets is not required if you are authenticating using a Run As account configured with a service principal.

For more information about checkpoints, see [Adding Checkpoints to a Script Workflow](#).

Next steps

- To learn about PowerShell Workflow runbooks, see [Tutorial: Create a PowerShell Workflow runbook](#).

Manage runbooks in Azure Automation

5/4/2021 • 12 minutes to read • [Edit Online](#)

You can add a runbook to Azure Automation by either creating a new one or importing an existing one from a file or the [Runbook Gallery](#). This article provides information for managing a runbook imported from a file. You can find all the details of accessing community runbooks and modules in [Runbook and module galleries for Azure Automation](#).

Create a runbook

Create a new runbook in Azure Automation using the Azure portal or Windows PowerShell. Once the runbook has been created, you can edit it using information in:

- [Edit textual runbook in Azure Automation](#)
- [Learn key Windows PowerShell Workflow concepts for Automation runbooks](#)
- [Manage Python 2 packages in Azure Automation](#)
- [Manage Python 3 packages \(preview\) in Azure Automation](#)

Create a runbook in the Azure portal

1. Sign in to the Azure [portal](#).
2. Search for and select **Automation Accounts**.
3. On the **Automation Accounts** page, select your Automation account from the list.
4. From the Automation account, select **Runbooks** under **Process Automation** to open the list of runbooks.
5. Click **Create a runbook**.
6. Enter a name for the runbook and select its [type](#). The runbook name must start with a letter and can contain letters, numbers, underscores, and dashes.
7. Click **Create** to create the runbook and open the editor.

Create a runbook with PowerShell

Use the `New-AzAutomationRunbook` cmdlet to create an empty runbook. Use the `Type` parameter to specify one of the runbook types defined for `New-AzAutomationRunbook`.

The following example shows how to create a new empty runbook.

```
$params = @{
    AutomationAccountName = 'MyAutomationAccount'
    Name                  = 'NewRunbook'
    ResourceGroupName     = 'MyResourceGroup'
    Type                 = 'PowerShell'
}
New-AzAutomationRunbook @params
```

Import a runbook

You can import a PowerShell or PowerShell Workflow (.ps1) script, a graphical runbook (.graphrunbook), or a Python 2 or Python 3 script (.py) to make your own runbook. You specify the [type of runbook](#) that is created during import, taking into account the following considerations.

- You can import a .ps1 file that doesn't contain a workflow into either a [PowerShell runbook](#) or a [PowerShell Workflow runbook](#). If you import it into a PowerShell Workflow runbook, it is converted to a

workflow. In this case, comments are included in the runbook to describe the changes made.

- You can import only a **.ps1** file containing a PowerShell Workflow into a [PowerShell Workflow runbook](#). If the file contains multiple PowerShell workflows, the import fails. You have to save each workflow to its own file and import each separately.
- Do not import a **.ps1** file containing a PowerShell Workflow into a [PowerShell runbook](#), as the PowerShell script engine can't recognize it.
- Only import a **.graphrunbook** file into a new [graphical runbook](#).

Import a runbook from the Azure portal

You can use the following procedure to import a script file into Azure Automation.

NOTE

You can only import a **.ps1** file into a PowerShell Workflow runbook using the portal.

1. In the Azure portal, search for and select **Automation Accounts**.
2. On the **Automation Accounts** page, select your Automation account from the list.
3. From the Automation account, select **Runbooks** under **Process Automation** to open the list of runbooks.
4. Click **Import a runbook**.
5. Click **Runbook file** and select the file to import.
6. If the **Name** field is enabled, you have the option of changing the runbook name. The name must start with a letter and can contain letters, numbers, underscores, and dashes.
7. The **runbook type** is automatically selected, but you can change the type after taking the applicable restrictions into account.
8. Click **Create**. The new runbook appears in the list of runbooks for the Automation account.
9. You have to [publish the runbook](#) before you can run it.

NOTE

After you import a graphical runbook, you can convert it to another type. However, you can't convert a graphical runbook to a textual runbook.

Import a runbook with Windows PowerShell

Use the [Import-AzAutomationRunbook](#) cmdlet to import a script file as a draft runbook. If the runbook already exists, the import fails unless you use the **Force** parameter with the cmdlet.

The following example shows how to import a script file into a runbook.

```
$params = @{
    AutomationAccountName = 'MyAutomationAccount'
    Name                  = 'Sample_TestRunbook'
    ResourceGroupName     = 'MyResourceGroup'
    Type                 = 'PowerShell'
    Path                 = 'C:\Runbooks\Sample_TestRunbook.ps1'
}
Import-AzAutomationRunbook @params
```

Handle resources

If your runbook creates a [resource](#), the script should check to see if the resource already exists before attempting to create it. Here's a basic example.

```

$vmName = 'WindowsVM1'
$rgName = 'MyResourceGroup'
$myCred = Get-AutomationPSCredential 'MyCredential'

$vmExists = Get-AzResource -Name $vmName -ResourceGroupName $rgName
if (-not $vmExists) {
    Write-Output "VM $vmName does not exist, creating"
    New-AzVM -Name $vmName -ResourceGroupName $rgName -Credential $myCred
} else {
    Write-Output "VM $vmName already exists, skipping"
}

```

Retrieve details from Activity log

You can retrieve runbook details, such as the person or account that started a runbook, from the [Activity log](#) for the Automation account. The following PowerShell example provides the last user to run the specified runbook.

```

$rgName = 'MyResourceGroup'
$accountName = 'MyAutomationAccount'
$runbookName = 'MyRunbook'
$startTime = (Get-Date).AddDays(-1)

$params = @{
    ResourceGroupName = $rgName
    StartTime        = $startTime
}
$JobActivityLogs = (Get-AzLog @params).Where( { $_.Authorization.Action -eq
'Microsoft.Automation/automationAccounts/jobs/write' })

$JobInfo = @{}
foreach ($log in $JobActivityLogs) {
    # Get job resource
    $JobResource = Get-AzResource -ResourceId $log.ResourceId

    if ($null -eq $JobInfo[$log.SubmissionTimestamp] -and $JobResource.Properties.Runbook.Name -eq
$runbookName) {
        # Get runbook
        $jobParams = @{
            ResourceGroupName      = $rgName
            AutomationAccountName = $accountName
            Id                   = $JobResource.Properties.JobId
        }
        $Runbook = Get-AzAutomationJob @jobParams | Where-Object RunbookName -EQ $runbookName

        # Add job information to hashtable
        $JobInfo.Add($log.SubmissionTimestamp, @{$Runbook.RunbookName, $Log.Caller,
$JobResource.Properties.jobId})
    }
}
$JobInfo.GetEnumerator() | Sort-Object Key -Descending | Select-Object -First 1

```

Track progress

It's a good practice to author your runbooks to be modular in nature, with logic that can be reused and restarted easily. Tracking progress in a runbook ensures that the runbook logic executes correctly if there are issues.

You can track the progress of a runbook by using an external source, such as a storage account, a database, or shared files. Create logic in your runbook to first check the state of the last action taken. Then, based on the results of the check, the logic can either skip or continue specific tasks in the runbook.

Prevent concurrent jobs

Some runbooks behave strangely if they run across multiple jobs at the same time. In this case, it's important for a runbook to implement logic to determine if there is already a running job. Here's a basic example.

```
# Authenticate to Azure
$connection = Get-AutomationConnection -Name AzureRunAsConnection
$cnParams = @{
    ServicePrincipal      = $true
    Tenant                = $connection.TenantId
    ApplicationId         = $connection.ApplicationId
    CertificateThumbprint = $connection.CertificateThumbprint
}
Connect-AzAccount @cnParams
$AzureContext = Set-AzContext -SubscriptionId $connection.SubscriptionID

# Check for already running or new runbooks
$runbookName = "RunbookName"
$rgName = "ResourceGroupName"
$accountName = "AutomationAccountName"
$jobs = Get-AzAutomationJob -ResourceGroupName $rgName -AutomationAccountName $accountName -RunbookName
$runbookName -AzContext $AzureContext

# Check to see if it is already running
$runningCount = ($jobs.Where( { $_.Status -eq 'Running' })).count

if (($jobs.Status -contains 'Running' -and $runningCount -gt 1 ) -or ($jobs.Status -eq 'New')) {
    # Exit code
    Write-Output "Runbook [$runbookName] is already running"
    exit 1
} else {
    # Insert Your code here
}
```

Alternatively, you can use PowerShell's splatting feature to pass the connection information to `Connect-AzAccount`. In that case, the first few lines of the previous sample would look like this.

```
# Authenticate to Azure
$connection = Get-AutomationConnection -Name AzureRunAsConnection
Connect-AzAccount @connection
$AzureContext = Set-AzContext -SubscriptionId $connection.SubscriptionID
```

For more information, see [about splatting](#).

Handle transient errors in a time-dependent script

Your runbooks must be robust and capable of handling [errors](#), including transient errors that can cause them to restart or fail. If a runbook fails, Azure Automation retries it.

If your runbook normally runs within a time constraint, have the script implement logic to check the execution time. This check ensures the running of operations such as startup, shutdown, or scale-out only during specific times.

NOTE

The local time on the Azure sandbox process is set to UTC. Calculations for date and time in your runbooks must take this fact into consideration.

Work with multiple subscriptions

Your runbook must be able to work with [subscriptions](#). For example, to handle multiple subscriptions, the runbook uses the [Disable-AzContextAutosave](#) cmdlet. This cmdlet ensures that the authentication context isn't retrieved from another runbook running in the same sandbox.

```
Disable-AzContextAutosave -Scope Process

$connection = Get-AutomationConnection -Name AzureRunAsConnection
$cnParams = @{
    ServicePrincipal      = $true
    Tenant                = $connection.TenantId
    ApplicationId         = $connection.ApplicationId
    CertificateThumbprint = $connection.CertificateThumbprint
}
Connect-AzAccount @cnParams

$childRunbookName = 'ChildRunbookDemo'
$accountName = 'MyAutomationAccount'
$rgName = 'MyResourceGroup'

$startParams = @{
    ResourceGroupName     = $rgName
    AutomationAccountName = $accountName
    Name                 = $childRunbookName
    DefaultProfile       = $AzureContext
}
Start-AzAutomationRunbook @startParams
```

Work with a custom script

NOTE

You can't normally run custom scripts and runbooks on the host with a Log Analytics agent installed.

To use a custom script:

1. Create an Automation account.
2. Deploy the [Hybrid Runbook Worker](#) role.
3. If on a Linux machine, you need elevated privileges. Sign in to [turn off signature checks](#).

Test a runbook

When you test a runbook, the [Draft version](#) is executed and any actions that it performs are completed. No job history is created, but the [output](#) and [warning and error](#) streams are displayed in the [Test output](#) pane.

Messages to the [verbose stream](#) are displayed in the Output pane only if the [VerbosePreference](#) variable is set to [Continue](#).

Even though the Draft version is being run, the runbook still executes normally and performs any actions against resources in the environment. For this reason, you should only test runbooks on non-production resources.

The procedure to test each [type of runbook](#) is the same. There's no difference in testing between the textual editor and the graphical editor in the Azure portal.

1. Open the Draft version of the runbook in either the [textual editor](#) or the [graphical editor](#).
2. Click **Test** to open the **Test** page.

3. If the runbook has parameters, they're listed in the left pane, where you can provide values to be used for the test.
4. If you want to run the test on a [Hybrid Runbook Worker](#), change **Run Settings** to **Hybrid Worker** and select the name of the target group. Otherwise, keep the default **Azure** to run the test in the cloud.
5. Click **Start** to begin the test.
6. You can use the buttons under the **Output** pane to stop or suspend a [PowerShell Workflow](#) or graphical runbook while it's being tested. When you suspend the runbook, it completes the current activity before being suspended. Once the runbook is suspended, you can stop it or restart it.
7. Inspect the output from the runbook in the **Output** pane.

Publish a runbook

When you create or import a new runbook, you have to publish it before you can run it. Each runbook in Azure Automation has a Draft version and a Published version. Only the Published version is available to be run, and only the Draft version can be edited. The Published version is unaffected by any changes to the Draft version. When the Draft version should be made available, you publish it, overwriting the current Published version with the Draft version.

Publish a runbook in the Azure portal

1. From the Azure portal, open the runbook in your Automation account.
2. Click **Edit**.
3. Click **Publish** and then **Yes** in response to the verification message.

Publish a runbook using PowerShell

Use the [Publish-AzAutomationRunbook](#) cmdlet to publish your runbook.

```
$accountName = "MyAutomationAccount"
$runbookName = "Sample_TestRunbook"
$rgName = "MyResourceGroup"

$publishParams = @{
    AutomationAccountName = $accountName
    ResourceGroupName      = $rgName
    Name                  = $runbookName
}
Publish-AzAutomationRunbook @publishParams
```

Schedule a runbook in the Azure portal

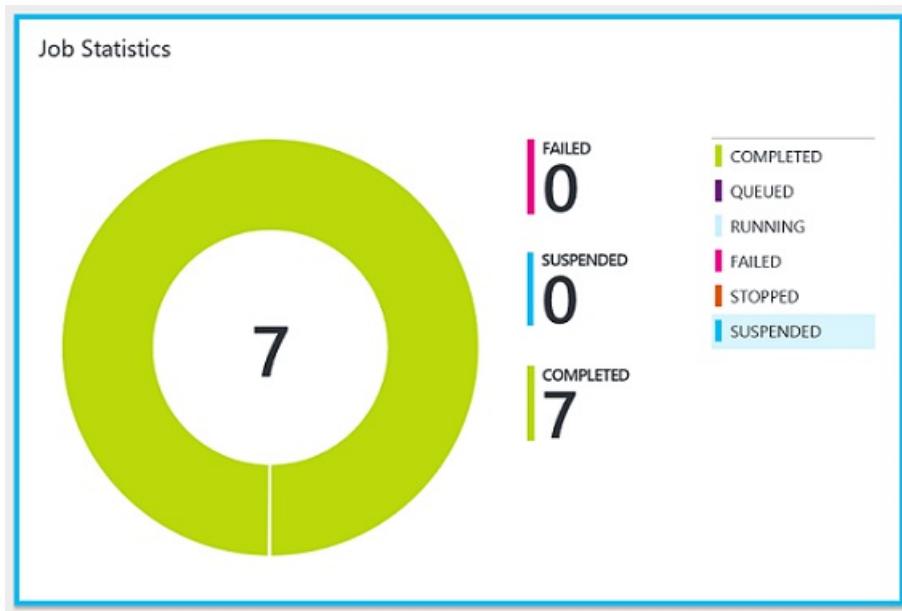
When your runbook has been published, you can schedule it for operation:

1. From the Azure portal, open the runbook in your Automation account.
2. Select **Schedules** under **Resources**.
3. Select **Add a schedule**.
4. In the Schedule Runbook pane, select **Link a schedule to your runbook**.
5. Choose **Create a new schedule** in the Schedule pane.
6. Enter a name, description, and other parameters in the New schedule pane.
7. Once the schedule is created, highlight it and click **OK**. It should now be linked to your runbook.
8. Look for an email in your mailbox to notify you of the runbook status.

Obtain job statuses

View statuses in the Azure portal

Details of job handling in Azure Automation are provided in [Jobs](#). When you are ready to see your runbook jobs, use Azure portal and access your Automation account. On the right, you can see a summary of all the runbook jobs in **Job Statistics**.



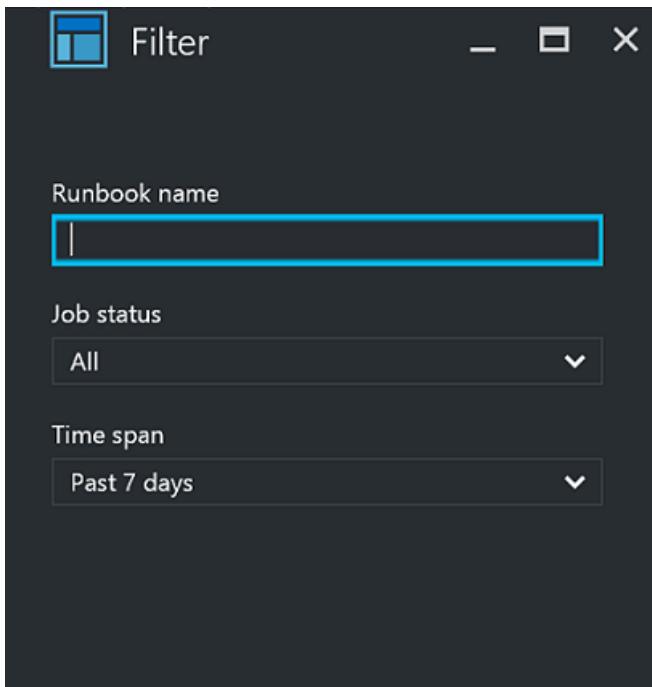
The summary displays a count and graphical representation of the job status for each job executed.

Clicking the tile presents the **Jobs** page, which includes a summarized list of all jobs executed. This page shows the status, runbook name, start time, and completion time for each job.

The table lists eight completed runbook jobs. Each row contains four columns: STATUS, RUNBOOK, CREATED, and LAST UPDATED. All jobs are marked as 'Completed' with a green checkmark icon.

STATUS	RUNBOOK	CREATED	LAST UPDATED
✓ Completed	Stop-AllVMs	11/1/2016 6:00 PM	11/1/2016 6:08 PM
✓ Completed	Stop-AllVMs	10/31/2016 6:00 PM	10/31/2016 6:12 PM
✓ Completed	Stop-AllVMs	10/30/2016 6:00 PM	10/30/2016 6:08 PM
✓ Completed	Stop-AllVMs	10/29/2016 6:00 PM	10/29/2016 6:09 PM
✓ Completed	Stop-AllVMs	10/28/2016 6:00 PM	10/28/2016 6:08 PM
✓ Completed	Stop-AllVMs	10/27/2016 6:00 PM	10/27/2016 6:08 PM
✓ Completed	Stop-AllVMs	10/26/2016 6:00 PM	10/26/2016 6:12 PM

You can filter the list of jobs by selecting **Filter jobs**. Filter on a specific runbook, job status, or a choice from the dropdown list, and provide the time range for the search.



Alternatively, you can view job summary details for a specific runbook by selecting that runbook from the Runbooks page in your Automation account and then selecting **Jobs**. This action presents the Jobs page. From here, you can click a job record to view its details and output.

The screenshot shows the Azure portal Jobs page. At the top, it displays the job title "Stop-AllVMs" and the creation time "11/1/2016 6:00 PM". Below the title, there are several actions: "Resume", "Stop", "Suspend", and "View source". The main area is titled "Overview" and contains the following information:

- Job Summary:** Job ID: edf36252-6cc9-4c59-92d7-dcaeb2285cac, Created: 11/1/2016 6:00 PM, Last updated: 11/1/2016 6:08 PM, Ran on Azure. Status: Completed (indicated by a green checkmark).
- Runbook:** Associated with the runbook "Stop-AllVMs".
- INPUT:** 0 items.
- Output:** 0 items.

Below the overview, there is a "Status" section with three categories: "Errors" (1), "Warnings" (0), and "All Logs". The "Errors" category is highlighted with a blue border. The "Exception" section below shows "None".

Retrieve job statuses using PowerShell

Use the [Get-AzAutomationJob](#) cmdlet to retrieve the jobs created for a runbook and the details of a particular job. If you start a runbook using [Start-AzAutomationRunbook](#), it returns the resulting job. Use [Get-AzAutomationJobOutput](#) to retrieve job output.

The following example gets the last job for a sample runbook and displays its status, the values provided for the runbook parameters, and the job output.

```

$getJobParams = @{
    AutomationAccountName = 'MyAutomationAccount'
    ResourceGroupName     = 'MyResourceGroup'
    Runbookname           = 'Test-Runbook'
}
$job = (Get-AzAutomationJob @getJobParams | Sort-Object LastModifiedDate -Desc)[0]
$job | Select-Object JobId, Status, JobParameters

$getOutputParams = @{
    AutomationAccountName = 'MyAutomationAccount'
    ResourceGroupName     = 'MyResourceGroup'
    Id                   = $job.JobId
    Stream               = 'Output'
}
Get-AzAutomationJobOutput @getOutputParams

```

The following example retrieves the output for a specific job and returns each record. If there's an [exception](#) for one of the records, the script writes the exception instead of the value. This behavior is useful since exceptions can provide additional information that might not be logged normally during output.

```

$params = @{
    AutomationAccountName = 'MyAutomationAccount'
    ResourceGroupName     = 'MyResourceGroup'
    Stream               = 'Any'
}
$output = Get-AzAutomationJobOutput @params

foreach ($item in $output) {
    $jobOutParams = @{
        AutomationAccountName = 'MyAutomationAccount'
        ResourceGroupName     = 'MyResourceGroup'
        Id                   = $item.StreamRecordId
    }
    $fullRecord = Get-AzAutomationJobOutputRecord @jobOutParams

    if ($fullRecord.Type -eq 'Error') {
        $fullRecord.Value.Exception
    } else {
        $fullRecord.Value
    }
}

```

Next steps

- To learn details of runbook management, see [Runbook execution in Azure Automation](#).
- To prepare a PowerShell runbook, see [Edit textual runbooks in Azure Automation](#).
- To troubleshoot issues with runbook execution, see [Troubleshoot runbook issues](#).

Edit textual runbooks in Azure Automation

7/21/2021 • 3 minutes to read • [Edit Online](#)

You can use the textual editor in Azure Automation to edit [PowerShell runbooks](#) and [PowerShell Workflow runbooks](#). This editor has the typical features of other code editors, such as IntelliSense. It also uses color coding with additional special features to assist you in accessing resources common to runbooks.

The textual editor includes a feature to insert code for cmdlets, assets, and child runbooks into a runbook. Instead of typing in the code yourself, you can select from a list of available resources and the editor inserts the appropriate code into the runbook.

Each runbook in Azure Automation has two versions, Draft and Published. You edit the Draft version of the runbook and then publish it so it can be executed. The Published version cannot be edited. For more information, see [Publish a runbook](#).

This article provides detailed steps for performing different functions with this editor. These are not applicable to [graphical runbooks](#). To work with these runbooks, see [Graphical authoring in Azure Automation](#).

IMPORTANT

Do not include the keyword "AzureRm" in any script designed to be executed with the Az module. Inclusion of the keyword, even in a comment, may cause the AzureRm to load and then conflict with the Az module.

Edit a runbook with the Azure portal

1. In the Azure portal, select your Automation account.
2. Under **PROCESS AUTOMATION**, select **Runbooks** to open the list of runbooks.
3. Choose the runbook to edit and then click **Edit**.
4. Edit the runbook.
5. Click **Save** when your edits are complete.
6. Click **Publish** if you want to publish the latest draft version of the runbook.

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 to use.
4. Right-click the cmdlet name to insert and select **Add to canvas**. If the cmdlet has more than one parameter set, the editor adds the default set. You can also expand the cmdlet to select a different parameter set.
5. Note that the code for the cmdlet is inserted with its entire list of parameters.
6. Provide an appropriate value in place of the value surrounded by angle brackets (<>) for any required parameters. Remove any parameters that you don't need.

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**.
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.

Insert an asset into a runbook

1. In the Canvas control 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 desired asset type.
4. Right-click the asset name to insert and select **Add to canvas**. For [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. Note that the code for the asset is inserted into the runbook.

Edit an Azure Automation runbook using Windows PowerShell

To edit a runbook with Windows PowerShell, use the editor of your choice and save the runbook to a **.ps1** file. You can use the [Export-AzAutomationRunbook](#) cmdlet to retrieve the contents of the runbook. You can use the [Import-AzAutomationRunbook](#) cmdlet to replace the existing draft runbook with the modified one.

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's also possible to retrieve the Published version of the runbook, although this version can't be changed.

```
$resourceGroupName = "MyResourceGroup"
$automationAccountName = "MyAutomationAccount"
$runbookName = "Hello-World"
$scriptFolder = "c:\runbooks"

Export-AzAutomationRunbook -Name $runbookName -AutomationAccountName $automationAccountName -
    ResourceGroupName $resourceGroupName -OutputFolder $scriptFolder -Slot Draft
```

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.

```
$resourceGroupName = "MyResourceGroup"
$automationAccountName = "MyAutomationAccount"
$runbookName = "Hello-World"
$scriptFolder = "c:\runbooks"

Import-AzAutomationRunbook -Path "$scriptFolder\Hello-World.ps1" -Name $runbookName -Type PowerShell -
    AutomationAccountName $automationAccountName -ResourceGroupName $resourceGroupName -Force
Publish-AzAutomationRunbook -Name $runbookName -AutomationAccountName $automationAccountName -
    ResourceGroupName $resourceGroupName
```

Next steps

- [Manage runbooks in Azure Automation](#).
- [Learning PowerShell workflow](#).
- [Graphical authoring in Azure Automation](#).
- [Certificates](#).
- [Connections](#).
- [Credentials](#).
- [Schedules](#).
- [Variables](#).

- [PowerShell cmdlet reference](#).

Author graphical runbooks in Azure Automation

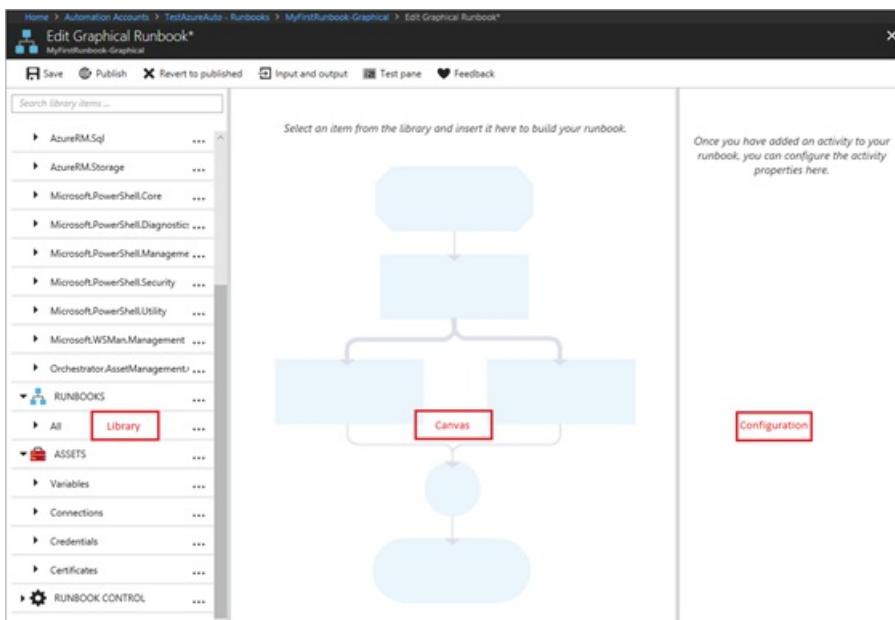
9/10/2021 • 22 minutes to read • [Edit Online](#)

All runbooks in Azure Automation are Windows PowerShell workflows. Graphical runbooks and graphical PowerShell Workflow runbooks generate PowerShell code that the Automation workers run but that you cannot view or modify. You can convert a graphical runbook to a graphical PowerShell Workflow runbook, and vice versa. However, you can't convert these runbooks to a textual runbook. Additionally, the Automation graphical editor can't import a textual runbook.

Graphical authoring allows you to create runbooks for Azure Automation without the complexities of the underlying Windows PowerShell or PowerShell Workflow code. You can add activities to the canvas from a library of cmdlets and runbooks, link them together, and configure them to form a workflow. If you have ever worked with System Center Orchestrator or Service Management Automation (SMA), graphical authoring should look familiar. This article provides an introduction to the concepts you need to get started creating a graphical runbook.

Overview of graphical editor

You can open the graphical editor in the Azure portal by creating or editing a graphical runbook.



The following sections describe the controls in the graphical editor.

Canvas control

The Canvas control allows you to design your runbook. You can add activities from the nodes in the Library control to the runbook and connect them with links to define runbook logic. At the bottom of the canvas, there are controls that allow you to zoom in and out.

Library control

The Library control allows you to select **activities** to add to your runbook. You add them to the canvas, where you can connect them to other activities. The Library control includes the sections defined in the following table.

SECTION	DESCRIPTION
Cmdlets	All the cmdlets that can be used in your runbook. Cmdlets are organized by module. All the modules that you have installed in your Automation account are available.
Runbooks	The runbooks in your Automation account. You can add these runbooks to the canvas to be used as child runbooks. Only runbooks of the same core type as the runbook being edited are shown. For graphical runbooks, only PowerShell-based runbooks are shown. For graphical PowerShell Workflow runbooks, only PowerShell Workflow-based runbooks are shown.
Assets	The automation assets in your Automation account that you can use in your runbook. Adding an asset to a runbook adds a workflow activity that gets the selected asset. In the case of variable assets, you can select whether to add an activity to get the variable or set the variable.
Runbook Control	Control activities that can be used in your current runbook. A Junction activity takes multiple inputs and waits until all have completed before continuing the workflow. A Code activity runs one or more lines of PowerShell or PowerShell Workflow code, depending on the graphical runbook type. You can use this activity for custom code or for functionality that is difficult to achieve with other activities.

Configuration control

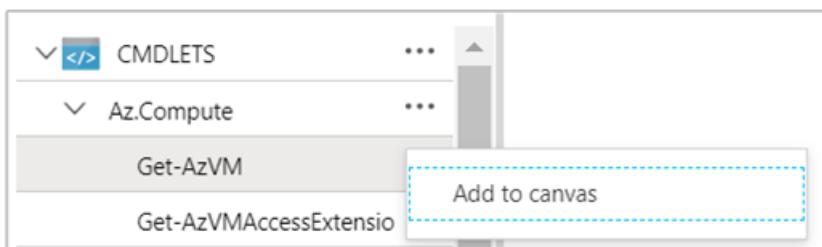
The Configuration control enables you to provide details for an object that is selected on the canvas. The properties available in this control depend on the type of object selected. When you choose an option in the Configuration control, it opens additional blades to provide more information.

Test control

The Test control is not displayed when the graphical editor is first started. It is opened when you interactively test a graphical runbook.

Use activities

Activities are the building blocks of a runbook. An activity can be a PowerShell cmdlet, a child runbook, or a workflow. You can add an activity to the runbook by right-clicking it in the Library control and selecting **Add to canvas**. You can then click and drag the activity to place it anywhere on the canvas that you like. The location of the activity on the canvas does not affect the operation of the runbook. You can lay out your runbook any way you find most suitable to visualize its operation.



Select an activity on the canvas to configure its properties and parameters in the Configuration blade. You can change the label of the activity to a name that you find descriptive. The runbook still runs the original cmdlet. You are simply changing the display name that the graphical editor uses. Note that the label must be unique within the runbook.

Parameter sets

A parameter set defines the mandatory and optional parameters that accept values for a particular cmdlet. All cmdlets have at least one parameter set, and some have several sets. If a cmdlet has multiple parameter sets, you must select the one to use before you can configure parameters. You can change the parameter set used by an activity by selecting **Parameter Set** and choosing another set. In this case, any parameter values that you have already configured are lost.

In the following example, the [Get-AzVM](#) cmdlet has three parameter sets. The example uses one set called **ListVirtualMachineInResourceGroupParamSet**, with a single optional parameter, for returning all virtual machines in a resource group. The example also uses the **GetVirtualMachineInResourceGroupParamSet** parameter set for specifying the virtual machine to return. This set has two mandatory parameters and one optional parameter.

The screenshot shows the 'Activity Parameter Configuration' dialog for an activity named 'Get-AzVM'. On the left, under 'Parameters', there is a section for 'Optional additional parameters' which includes a 'Configure parameters' button. On the right, the 'Parameter Set' section is open, displaying three available sets:

- ListLocationVirtualMachinesParamSet**: Includes parameters: -Status <System.Management.Automation.SwitchParameter>, -Location <System.String>, -DefaultProfile <Microsoft.Azure.Commands.Common.Authentication.Abstractions.Core.IAzureContextC...>.
- ListNextLinkVirtualMachinesParamSet**: Includes parameters: -Status <System.Management.Automation.SwitchParameter>, -NextLink <System.Uri>, -DefaultProfile <Microsoft.Azure.Commands.Common.Authentication.Abstractions.Core.IAzureContextC...>.
- GetVirtualMachineInResourceGroupParamSet**: Includes parameters: -Name <System.String>, -DefaultProfile <Microsoft.Azure.Commands.Common.Authentication.Abstractions.Core.IAzureContextC...>, -ResourceGroupName <System.String>, -DisplayHint <Microsoft.Azure.Commands.Compute.Models.DisplayHintType>, -Status <System.Management.Automation.SwitchParameter>.

Below these sets is a 'DefaultParamSet' section with parameters: [-Status <System.Management.Automation.SwitchParameter>], [-DefaultProfile <Microsoft.Azure.Commands.Common.Authentication.Abstractions.Core.IAzureContextC...>], [-ResourceGroupName <System.String>], [-Name <System.String>].

Parameter values

When you specify a value for a parameter, you select a data source to determine how the value is specified. The data sources that are available for a particular parameter depend on the valid values for that parameter. For example, Null is not an available option for a parameter that does not allow null values.

DATA SOURCE	DESCRIPTION
Constant Value	Type in a value for the parameter. This data source is only available for the following data types: Int32, Int64, String, Boolean, DateTime, Switch.
Activity Output	Use output from an activity that precedes the current activity in the workflow. All valid activities are listed. For the parameter value, use just the activity that produces the output. If the activity outputs an object with multiple properties, you can type in the name of a specific property after selecting the activity.
Runbook Input	Select a runbook input as an input for the activity parameter.
Variable Asset	Select an Automation variable as input.
Credential Asset	Select an Automation credential as input.
Certificate Asset	Select an Automation certificate as input.

DATA SOURCE	DESCRIPTION
Connection Asset	Select an Automation connection as input.
PowerShell Expression	Specify a simple PowerShell expression . The expression is evaluated before the activity and the result is used for the parameter value. You can use variables to refer to the output of an activity or a runbook input parameter.
Not Configured	Clear any value that was previously configured.

Optional additional parameters

All cmdlets have the option to provide additional parameters. These are PowerShell-common parameters or other custom parameters. The graphical editor presents a text box where you can provide parameters using PowerShell syntax. For example, to use the `-Verbose` common parameter, you should specify `-Verbose:$True`.

Retry activity

Retry functionality for an activity allows it to be run multiple times until a particular condition is met, much like a loop. You can use this feature for activities that should run multiple times, are error prone, might need more than one attempt for success, or test the output information of the activity for valid data.

When you enable retry for an activity, you can set a delay and a condition. The delay is the time (measured in seconds or minutes) that the runbook waits before it runs the activity again. If you don't specify a delay, the activity runs again immediately after it completes.

Enable retry ⓘ
 Yes No
 Delay before each retry attempt ⓘ
 30 Seconds

The retry condition is a PowerShell expression that is evaluated after each time that the activity runs. If the expression resolves to True, the activity runs again. If the expression resolves to False, the activity does not run again and the runbook moves on to the next activity.

Retry until this condition is true ⓘ
 \$RetryData.NumberOfAttempts -ge 10
 Examples that can be used in retry condition: ⓘ
 \$RetryData.NumberOfAttempts -ge 10 # retry 10 times
 \$RetryData.Output.Count -ge 1 # retry until output is produced
 \$RetryData.TotalDuration.TotalMinutes -ge 2 # retry for maximum of 2 minutes

The retry condition can use a variable named `$RetryData` that provides access to information about the activity retries. This variable has the properties in the following table:

PROPERTY	DESCRIPTION
<code>NumberOfAttempts</code>	Number of times that the activity has been run.

PROPERTY	DESCRIPTION
Output	Output from the last run of the activity.
TotalDuration	Timed elapsed since the activity was started the first time.
StartedAt	Time (in UTC format) when the activity was first started.

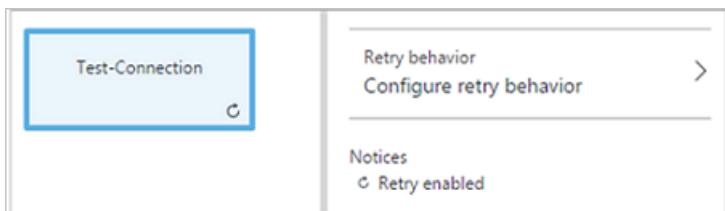
The following are examples of activity retry conditions.

```
# Run the activity exactly 10 times.
$RetryData.NumberOfAttempts -ge 10
```

```
# Run the activity repeatedly until it produces any output.
$RetryData.Output.Count -ge 1
```

```
# Run the activity repeatedly until 2 minutes has elapsed.
$RetryData.TotalDuration.TotalMinutes -ge 2
```

After you configure a retry condition for an activity, the activity includes two visual cues to remind you. One is presented in the activity and the other is shown when you review the configuration of the activity.



Workflow Script control

A workflow Script control is a special activity that accepts PowerShell or PowerShell Workflow script, depending on the type of graphical runbook being authored. This control provides functionality that might not be available by other means. It cannot accept parameters, but it can use variables for activity output and runbook input parameters. Any output of the activity is added to the databus. An exception is output with no outgoing link, in which case the output is added to the output of the runbook.

For example, the following code performs date calculations using a runbook input variable named `NumberOfDays`. It then sends a calculated DateTime value as output to be used by subsequent activities in the runbook.

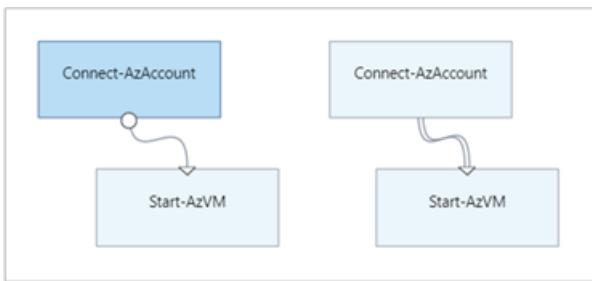
```
$DateTimeNow = (Get-Date).ToUniversalTime()
$DateTimeStart = ($DateTimeNow).AddDays(-$NumberOfDays)
$DateTimeStart
```

Use links for workflow

A link in a graphical runbook connects two activities. It is displayed on the canvas as an arrow pointing from the source activity to the destination activity. The activities run in the direction of the arrow with the destination activity starting after the source activity completes.

Create a link

You can create a link between two activities by selecting the source activity and clicking the circle at the bottom of the shape. Drag the arrow to the destination activity and release.



Select the link to configure its properties in the Configuration blade. Properties include the link type, which is described in the following table.

LINK TYPE	DESCRIPTION
Pipeline	The destination activity runs once for each object output from the source activity. The destination activity does not run if the source activity results in no output. Output from the source activity is available as an object.
Sequence	The destination activity runs only once when it receives output from the source activity. Output from the source activity is available as an array of objects.

Start runbook activity

A graphical runbook starts with any activities that do not have an incoming link. There is often only one activity that acts as the starting activity for the runbook. If multiple activities do not have an incoming link, the runbook starts by running them in parallel. It follows the links to run other activities as each completes.

Specify link conditions

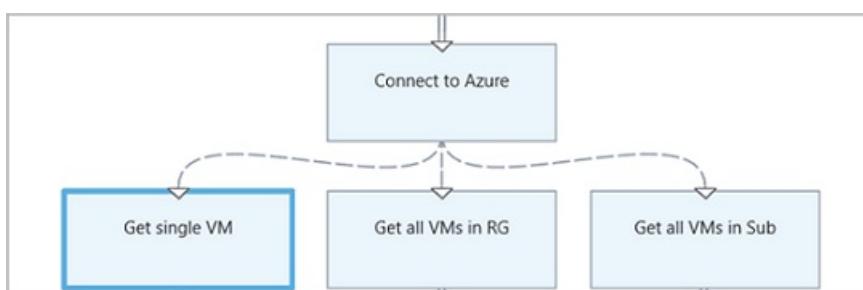
When you specify a condition on a link, the destination activity runs only if the condition resolves to True. You typically use an `ActivityOutput` variable in a condition to retrieve the output from the source activity.

For a pipeline link, you must specify a condition for a single object. The runbook evaluates the condition for each object output by the source activity. It then runs the destination activity for each object that satisfies the condition. For example, with a source activity of `Get-AzVM`, you can use the following syntax for a conditional pipeline link to retrieve only virtual machines in the resource group named Group1.

```
$ActivityOutput['Get Azure VMs'].Name -match "Group1"
```

For a sequence link, the runbook only evaluates the condition once, since a single array containing all objects from the source activity is returned. Because of this, the runbook can't use a sequence link for filtering, like it can with a pipeline link. The sequence link can simply determine whether or not the next activity is run.

For example, take the following set of activities in our **Start VM** runbook:



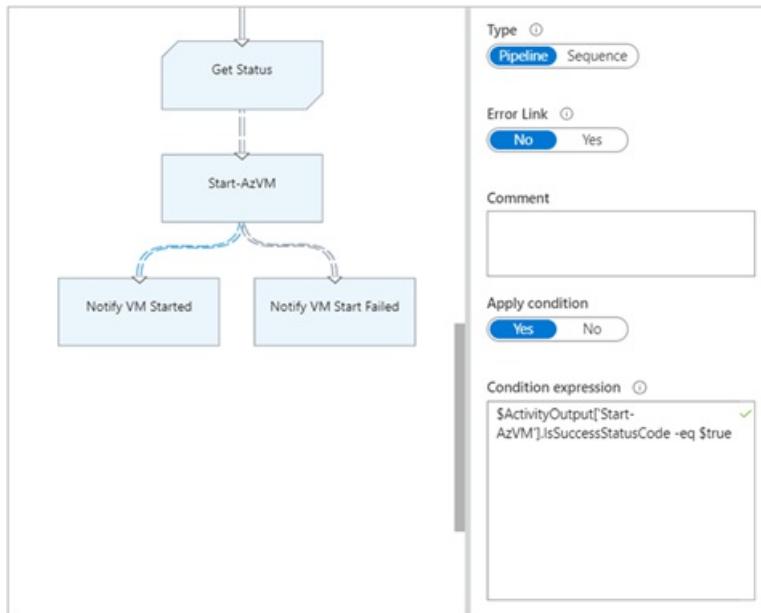
The runbook uses three different sequence links that verify values of the input parameters `VMName` and `ResourceGroupName` to determine the appropriate action to take. Possible actions are start a single VM, start all

VMs in the resource group, or start all VMs in a subscription. For the sequence link between [Connect to Azure](#) and [Get single VM](#), here is the condition logic:

```
<#
Both VMName and ResourceGroupName runbook input parameters have values
#>
(
((($VMName -ne $null) -and ($VMName.Length -gt 0))
 ) -and (
(($ResourceGroupName -ne $null) -and ($ResourceGroupName.Length -gt 0))
 )
```

When you use a conditional link, the data available from the source activity to other activities in that branch is filtered by the condition. If an activity is the source to multiple links, the data available to activities in each branch depends on the condition in the link connecting to that branch.

For example, the [Start-AzVM](#) activity in the runbook below starts all virtual machines. It has two conditional links. The first conditional link uses the expression `$ActivityOutput['Start-AzVM'].IsSuccessStatusCode -eq $true` to filter if the [Start-AzVM](#) activity completes successfully. The second conditional link uses the expression `$ActivityOutput['Start-AzVM'].IsSuccessStatusCode -ne $true` to filter if the [Start-AzVm](#) activity fails to start the virtual machine.



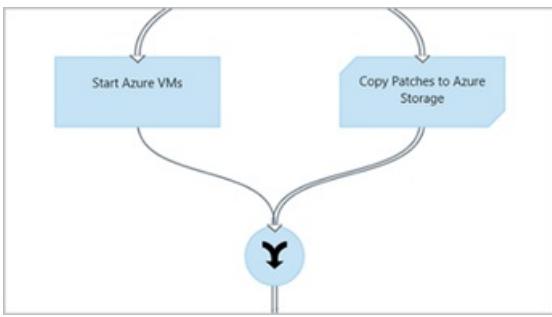
Any activity that follows the first link and uses the activity output from [Get-AzureVM](#) only retrieves the virtual machines that were started at the time when [Get-AzureVM](#) was run. Any activity that follows the second link only gets the virtual machines that were stopped at the time when [Get-AzureVM](#) was run. Any activity following the third link gets all virtual machines regardless of their running state.

Use junctions

A junction is a special activity that waits until all incoming branches have completed. This allows the runbook to run multiple activities in parallel and ensure that all have completed before moving on.

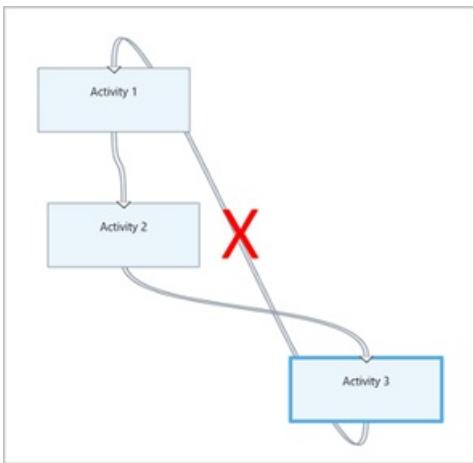
While a junction can have an unlimited number of incoming links, only one of those links can be a pipeline. The number of incoming sequence links is not constrained. You can create the junction with multiple incoming pipeline links and save the runbook, but it will fail when it is run.

The example below is part of a runbook that starts a set of virtual machines while simultaneously downloading patches to be applied to those machines. It uses a junction to ensure that both processes are completed before the runbook continues.



Work with cycles

A cycle is formed when a destination activity links back to its source activity or to another activity that eventually links back to its source. Graphical authoring does not currently support cycles. If your runbook has a cycle, it saves properly but receives an error when it runs.



Share data between activities

Any data that an activity outputs with an outgoing link is written to the databus for the runbook. Any activity in the runbook can use data on the databus to populate parameter values or include in script code. An activity can access the output of any previous activity in the workflow.

How the data is written to the databus depends on the type of link on the activity. For a pipeline link, the data is output as multiple objects. For a sequence link, the data is output as an array. If there is only one value, it is output as a single-element array.

Your runbook has two ways to access data on the databus:

- Use an activity output data source.
- Use a PowerShell expression data source.

The first mechanism uses an activity output data source to populate a parameter of another activity. If the output is an object, the runbook can specify a single property.

The second data access mechanism retrieves the output of an activity in a PowerShell expression data source or a workflow script activity with an `ActivityOutput` variable, using the syntax shown below. If the output is an object, your runbook can specify a single property.

```
$ActivityOutput['Activity Label']
$ActivityOutput['Activity Label'].PropertyName
```

Use checkpoints

You can set [checkpoints](#) in a graphical PowerShell Workflow runbook by selecting **Checkpoint runbook** on any activity. This causes a checkpoint to be set after the activity runs.



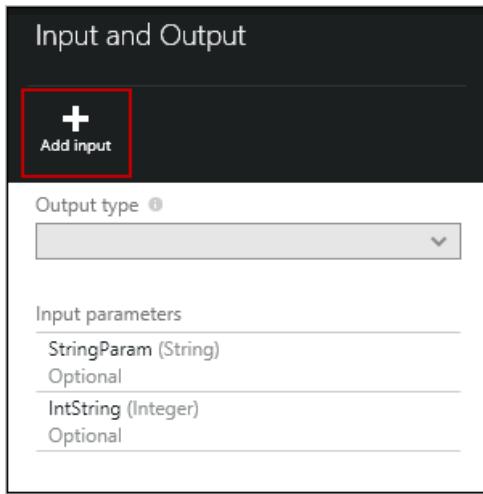
Checkpoints are only enabled in graphical PowerShell Workflow runbooks, and are not available in graphical runbooks. If the runbook uses Azure cmdlets, it should follow any checkpointed activity with a `Connect-AzAccount` activity. The connect operation is used in case the runbook is suspended and must restart from this checkpoint on a different worker.

Handle runbook input

A runbook requires input either from a user starting the runbook through the Azure portal or from another runbook, if the current one is used as a child. For example, for a runbook that creates a virtual machine, the user might need to provide such information as the name of the virtual machine and other properties each time the runbook starts.

The runbook accepts input by defining one or more input parameters. The user provides values for these parameters each time the runbook starts. When the user starts the runbook using the Azure portal, the user is prompted to provide values for each input parameter supported by the runbook.

When authoring your runbook, you can access its input parameters by clicking **Input and output** on the runbook toolbar. This opens the Input and Output control where you can edit an existing input parameter or create a new one by clicking **Add input**.



Each input parameter is defined by the properties in the following table:

PROPERTY	DESCRIPTION
Name	Required. The name of the parameter. The name must be unique within the runbook. It must start with a letter and can contain only letters, numbers, and underscores. The name cannot contain a space.
Description	Optional. Description of the purpose for the input parameter.
Type	Optional. Data type expected for the parameter value. The Azure portal provides an appropriate control for the data type for each parameter when prompting for input. Supported parameter types are String, Int32, Int64, Decimal, Boolean, DateTime, and Object. If a data type is not selected, it defaults to String.
Mandatory	Optional. Setting that specifies if a value must be provided for the parameter. If you choose <code>yes</code> , a value must be provided when the runbook is started. If you choose <code>no</code> , a value is not required when the runbook is started, and a default value can be used. The runbook cannot start if you do not provide a value for each mandatory parameter that does not have a default value defined.
Default Value	Optional. The value used for a parameter if one is not passed in when the runbook is started. To set a default value, choose <code>Custom</code> . Select <code>None</code> if you don't want to provide any default value.

Handle runbook output

Graphical authoring saves data created by any activity that does not have an outgoing link to the [output of the runbook](#). The output is saved with the runbook job and is available to a parent runbook when the runbook is used as a child.

Work with PowerShell expressions

One of the advantages of graphical authoring is that it allows you to build a runbook with minimal knowledge of PowerShell. Currently, though, you do need to know a bit of PowerShell for populating certain [parameter values](#) and for setting [link conditions](#). This section provides a quick introduction to PowerShell expressions. Full

details of PowerShell are available at [Scripting with Windows PowerShell](#).

Use a PowerShell expression as a data source

You can use a PowerShell expression as a data source to populate the value of an [activity parameter](#) with the results of PowerShell code. The expression can be a single line of code that performs a simple function or multiple lines that perform some complex logic. Any output from a command that is not assigned to a variable is output to the parameter value.

For example, the following command outputs the current date.

```
Get-Date
```

The next code snippet builds a string from the current date and assigns it to a variable. The code sends the contents of the variable to the output.

```
$string = "The current date is " + (Get-Date)  
$string
```

The following commands evaluate the current date and return a string indicating whether the current day is a weekend or a weekday.

```
$date = Get-Date  
if (($date.DayOfWeek = "Saturday") -or ($date.DayOfWeek = "Sunday")) { "Weekend" }  
else { "Weekday" }
```

Use activity output

To use the output from a previous activity in your runbook, use the `ActivityOutput` variable with the following syntax.

```
$ActivityOutput['Activity Label'].PropertyName
```

For example, you can have an activity with a property that requires the name of a virtual machine. In this case, your runbook can use the following expression.

```
$ActivityOutput['Get-AzureVM'].Name
```

If the property requires the virtual machine object instead of just a name, the runbook returns the entire object using the following syntax.

```
$ActivityOutput['Get-AzureVM']
```

The runbook can use the output of an activity in a more complex expression, such as the following. This expression concatenates text to the virtual machine name.

```
"The computer name is " + $ActivityOutput['Get-AzureVM'].Name
```

Compare values

Use [comparison operators](#) to compare values or determine if a value matches a specified pattern. A comparison returns a value of either True or False.

For example, the following condition determines if the virtual machine from an activity named `Get-AzureVM` is currently stopped.

```
$ActivityOutput["Get-AzureVM"].PowerState -eq "Stopped"
```

The following condition determines if the same virtual machine is in any state other than stopped.

```
$ActivityOutput["Get-AzureVM"].PowerState -ne "Stopped"
```

You can join multiple conditions in your runbook using a [logical operator](#), such as `-and` or `-or`. For example, the following condition checks to see if the virtual machine in the previous example is in a state of Stopped or Stopping.

```
($ActivityOutput["Get-AzureVM"].PowerState -eq "Stopped") -or ($ActivityOutput["Get-AzureVM"].PowerState -eq "Stopping")
```

Use hashtables

[Hashtables](#) are name-value pairs that are useful for returning a set of values. You might also see a hashtable referred to as a dictionary. Properties for certain activities expect a hashtable instead of a simple value.

Create a hashtable using the following syntax. It can contain any number of entries, but each is defined by a name and value.

```
@{ <name> = <value>; [<name> = <value> ] ... }
```

For example, the following expression creates a hashtable to be used as the data source for an activity parameter that expects a hashtable of values for an internet search.

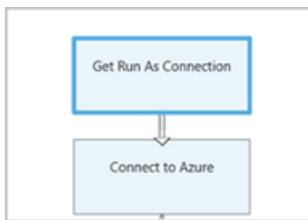
```
$query = "Azure Automation"
$count = 10
$h = @{'q'=$query; 'lr'='lang_ja'; 'count'=$Count}
$h
```

The following example uses output from an activity called `Get Twitter Connection` to populate a hashtable.

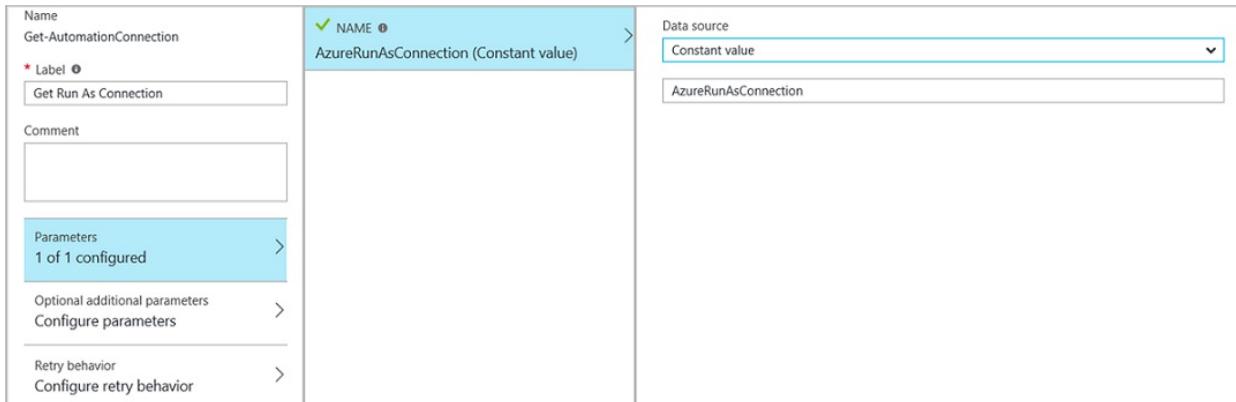
```
@{ 'ApiKey'=$ActivityOutput['Get Twitter Connection'].ConsumerAPIKey;
  'ApiSecret'=$ActivityOutput['Get Twitter Connection'].ConsumerAPISecret;
  'AccessToken'=$ActivityOutput['Get Twitter Connection'].AccessToken;
  'AccessTokenSecret'=$ActivityOutput['Get Twitter Connection'].AccessTokenSecret}
```

Authenticate to Azure resources

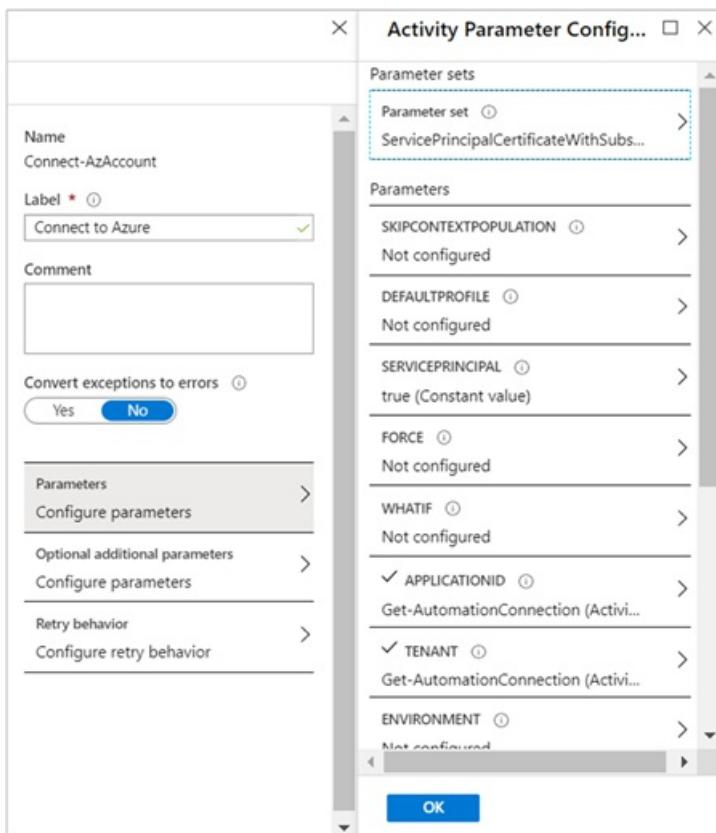
Runbooks in Azure Automation that manage Azure resources require authentication to Azure. The [Run As account](#), also referred to as a service principal, is the default mechanism that an Automation runbook uses to access Azure Resource Manager resources in your subscription. You can add this functionality to a graphical runbook by adding the `AzureRunAsConnection` connection asset, which uses the PowerShell `Get-AutomationConnection` cmdlet, to the canvas. You can also add the `Connect-AzAccount` cmdlet. This scenario is illustrated in the following example.



The `Get Run As Connection` activity, or `Get-AutomationConnection`, is configured with a constant value data source named `AzureRunAsConnection`.



The next activity, `Connect-AzAccount`, adds the authenticated Run As account for use in the runbook.



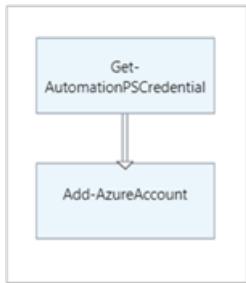
NOTE

For PowerShell runbooks, `Add-AzAccount` and `Add-AzureRMAccount` are aliases for `Connect-AzAccount`. Note that these aliases are not available for your graphical runbooks. A graphical runbook can only use `Connect-AzAccount` itself.

For the parameter fields `APPLICATIONID`, `CERTIFICATETHUMBPRINT`, and `TENANTID`, specify the name of the property for the field path, since the activity outputs an object with multiple properties. Otherwise, when the runbook executes, it fails while attempting to authenticate. This is what you need at a minimum to authenticate your runbook with the Run As account.

Some subscribers create an Automation account using an [Azure AD user account](#) to manage Azure classic deployment or for Azure Resource Manager resources. To maintain backward compatibility for these subscribers, the authentication mechanism to use in your runbook is the `Add-AzureAccount` cmdlet with a [credential asset](#). The asset represents an Active Directory user with access to the Azure account.

You can enable this functionality for your graphical runbook by adding a credential asset to the canvas, followed by an `Add-AzureAccount` activity that uses the credential asset for its input. See the following example.



The runbook must authenticate at its start and after each checkpoint. Thus you must use an `Add-AzureAccount` activity after any `Checkpoint-Workflow` activity. You do not need to use an additional credential activity.

CREDENTIAL ⓘ Get-AutomationPSCredential (Activity output)	>	X UNSELECT ▶ ACTIVITY OUTPUT	>	Activity output ▶ Get-AutomationPSCredential
ENVIRONMENT ⓘ Not configured	>	▶ POWERSHELL EXPRESSION	>	Field path []
PROFILE ⓘ Not configured	>			

Export a graphical runbook

You can only export the published version of a graphical runbook. If the runbook has not yet been published, the **Export** button is disabled. When you click the **Export** button, the runbook downloads to your local computer. The name of the file matches the name of the runbook with a **.graphrunbook** extension.

Import a graphical runbook

You can import a graphical or graphical PowerShell Workflow runbook file by selecting the **Import** option when adding a runbook. When you select the file to import, you can keep the same name or provide a new one. The **Runbook Type** field displays the type of runbook after it assesses the file selected. If you attempt to select a different type that is not correct, the graphical editor presents a message noting that there are potential conflicts and there might be syntax errors during conversion.



Test a graphical runbook

Each graphical runbook in Azure Automation has a Draft version and a Published version. You can run only the Published version, while you can only edit the Draft version. The Published version is unaffected by any changes to the Draft version. When the Draft version is ready for use, you publish it, which overwrites the current Published version with your Draft version.

You can test the Draft version of a runbook in the Azure portal while leaving the Published version unchanged. Alternatively, you can test a new runbook before it has been published so that you can verify that the runbook works correctly before any version replacements. Testing of a runbook executes the Draft version and ensures that any actions that it performs are completed. No job history is created, but the Test Output pane displays the output.

Open the Test control for your graphical runbook by opening the runbook for edit and then clicking **Test pane**. The Test control prompts for input parameters, and you can start the runbook by clicking **Start**.

Publish a graphical runbook

Publish a graphical runbook by opening the runbook for editing and then clicking **Publish**. Possible statuses for the runbook are:

- New -- the runbook has not been published yet.
- Published -- the runbook has been published.
- In edit -- the runbook has been edited after it has been published, and the Draft and Published versions are different.

NAME	AUTHORING STATUS	LAST MODIFIED	TAGS
 Hello-World	 Published	12/15/2017 4:08 PM	
 MyFirstRunbook-Graphical	 In edit	12/6/2017 8:02 AM	
 MyFirstRunbook-PowerShell	 Published	1/24/2018 12:04 PM	
 MyFirstRunbook-Python	 New	1/24/2018 12:32 PM	

You have the option to revert to the Published version of a runbook. This operation throws away any changes made since the runbook was last published. It replaces the Draft version of the runbook with the Published version.

Next steps

- To get started with graphical runbooks, see [Tutorial: Create a graphical runbook](#).
- To know more about runbook types and their advantages and limitations, see [Azure Automation runbook types](#).
- To understand how to authenticate using the Automation Run As account, see [Run As account](#).
- For a PowerShell cmdlet reference, see [Az.Automation](#).

Create modular runbooks in Automation

9/13/2021 • 6 minutes to read • [Edit Online](#)

It's a recommended practice in Azure Automation to write reusable, modular runbooks with a discrete function that is called by other runbooks. A parent runbook often calls one or more child runbooks to perform required functionality.

There are two ways to call a child runbook, and there are distinct differences that you should understand to determine which is best for your scenario(s). The following table summarizes the differences between the two ways to call one runbook from another.

	INLINE	CMDLET
Job	Child runbooks run in the same job as the parent.	A separate job is created for the child runbook.
Execution	Parent runbook waits for the child runbook to complete before continuing.	Parent runbook continues immediately after child runbook is started <i>or</i> parent runbook waits for the child job to finish.
Output	Parent runbook can directly get output from child runbook.	Parent runbook must retrieve output from child runbook job <i>or</i> parent runbook can directly get output from child runbook.
Parameters	Values for the child runbook parameters are specified separately and can use any data type.	Values for the child runbook parameters have to be combined into a single hashtable. This hashtable can only include simple, array, and object data types that use JSON serialization.
Automation Account	Parent runbook can only use child runbook in the same Automation account.	Parent runbooks can use a child runbook from any Automation account, from the same Azure subscription, and even from a different subscription to which you have a connection.
Publishing	Child runbook must be published before parent runbook is published.	Child runbook is published anytime before parent runbook is started.

Invoke a child runbook using inline execution

To invoke a runbook inline from another runbook, use the name of the runbook and provide values for its parameters, just like you would use an activity or a cmdlet. All runbooks in the same Automation account are available to all others to be used in this manner. The parent runbook waits for the child runbook to complete before moving to the next line, and any output returns directly to the parent.

When you invoke a runbook inline, it runs in the same job as the parent runbook. There's no indication in the job history of the child runbook. Any exceptions and any stream outputs from the child runbook are associated with the parent. This behavior results in fewer jobs and makes them easier to track and to troubleshoot.

When a runbook is published, any child runbooks that it calls must already be published. The reason is that Azure Automation builds an association with any child runbooks when it compiles a runbook. If the child runbooks haven't already been published, the parent runbook appears to publish properly but generates an exception when it's started. If this happens, you can republish the parent runbook to properly reference the child runbooks. You don't need to republish the parent runbook if any child runbook is changed because the association has already been created.

The parameters of a child runbook called inline can be of any data type, including complex objects. There's no [JSON serialization](#), as there is when you start the runbook using the Azure portal or with the [Start-AzAutomationRunbook](#) cmdlet.

Runbook types

Which runbook types can call each other?

- A [PowerShell runbook](#) and a [graphical runbook](#) can call each other inline, as both are PowerShell-based.
- A [PowerShell Workflow runbook](#) and a graphical PowerShell Workflow runbook can call each other inline, as both are PowerShell Workflow-based.
- The PowerShell types and the PowerShell Workflow types can't call each other inline, and must use [Start-AzAutomationRunbook](#).

When does publish order matter?

The publish order of runbooks only matters for PowerShell Workflow and graphical PowerShell Workflow runbooks.

When your runbook calls a graphical or PowerShell Workflow child runbook using inline execution, it uses the name of the runbook. The name must start with `.\\` to specify that the script is located in the local directory.

Example

The following example starts a test child runbook that accepts a complex object, an integer value, and a boolean value. The output of the child runbook is assigned to a variable. In this case, the child runbook is a PowerShell Workflow runbook.

```
$vm = Get-AzVM -ResourceGroupName "LabRG" -Name "MyVM"
$output = PSWF-ChildRunbook -VM $vm -RepeatCount 2 -Restart $true
```

Here's the same example using a PowerShell runbook as the child.

```
$vm = Get-AzVM -ResourceGroupName "LabRG" -Name "MyVM"
$output = .\PS-ChildRunbook.ps1 -VM $vm -RepeatCount 2 -Restart $true
```

Start a child runbook using a cmdlet

IMPORTANT

If your runbook invokes a child runbook with the [Start-AzAutomationRunbook](#) cmdlet with the [Wait](#) parameter and the child runbook produces an object result, the operation might encounter an error. To work around the error, see [Child runbooks with object output](#) to learn how to implement the logic to poll for the results using the [Get-AzAutomationJobOutputRecord](#) cmdlet.

You can use [Start-AzAutomationRunbook](#) to start a runbook as described in [To start a runbook with Windows PowerShell](#). There are two modes of use for this cmdlet. In one mode, the cmdlet returns the job ID when the job is created for the child runbook. In the other mode, which your script enables by specifying the [Wait](#) parameter,

the cmdlet waits until the child job finishes and returns the output from the child runbook.

The job from a child runbook started with a cmdlet runs separately from the parent runbook job. This behavior results in more jobs than starting the runbook inline, and makes the jobs more difficult to track. The parent can start more than one child runbook asynchronously without waiting for each to complete. For this parallel execution calling the child runbooks inline, the parent runbook must use the [parallel keyword](#).

Child runbook output doesn't return to the parent runbook reliably because of timing. Also, variables such as `$VerbosePreference`, `$WarningPreference`, and others might not be propagated to the child runbooks. To avoid these issues, you can start the child runbooks as separate Automation jobs using `Start-AzAutomationRunbook` with the `Wait` parameter. This technique blocks the parent runbook until the child runbook is complete.

If you don't want the parent runbook to be blocked on waiting, you can start the child runbook using `Start-AzAutomationRunbook` without the `Wait` parameter. In this case, your runbook must use [Get-AzAutomationJob](#) to wait for job completion. It must also use [Get-AzAutomationJobOutput](#) and [Get-AzAutomationJobOutputRecord](#) to retrieve the results.

Parameters for a child runbook started with a cmdlet are provided as a hashtable, as described in [Runbook parameters](#). Only simple data types can be used. If the runbook has a parameter with a complex data type, then it must be called inline.

The subscription context might be lost when starting child runbooks as separate jobs. For the child runbook to execute Az module cmdlets against a specific Azure subscription, the child must authenticate to this subscription independently of the parent runbook.

If jobs within the same Automation account work with more than one subscription, selecting a subscription in one job can change the currently selected subscription context for other jobs. To avoid this situation, use `Disable-AzContextAutosave -Scope Process` at the beginning of each runbook. This action only saves the context to that runbook execution.

Example

The following example starts a child runbook with parameters and then waits for it to complete using the `Start-AzAutomationRunbook` cmdlet with the `Wait` parameter. Once completed, the example collects cmdlet output from the child runbook. To use `Start-AzAutomationRunbook`, the script must authenticate to your Azure subscription.

```
# Ensure that the runbook does not inherit an AzContext
Disable-AzContextAutosave -Scope Process

# Connect to Azure with user-assigned managed identity
Connect-AzAccount -Identity
$identity = Get-AzUserAssignedIdentity -ResourceGroupName <ResourceGroupName> -Name
<UserAssignedManagedIdentity>
Connect-AzAccount -Identity -AccountId $identity.ClientId

$AzureContext = Set-AzContext -SubscriptionId ($identity.id -split "/")[2]

$params = @{"VMName"="MyVM"; "RepeatCount"=2; "Restart"=$true}

Start-AzAutomationRunbook ` 
    -AutomationAccountName 'MyAutomationAccount' ` 
    -Name 'Test-ChildRunbook' ` 
    -ResourceGroupName 'LabRG' ` 
    -AzContext $AzureContext ` 
    -Parameters $params -Wait
```

Next steps

- To run your runbook, see [Start a runbook in Azure Automation](#).
- For monitoring of runbook operation, see [Runbook output and messages in Azure Automation](#).

Configure runbook input parameters in Automation

9/13/2021 • 11 minutes to read • [Edit Online](#)

Runbook input parameters increase the flexibility of a runbook by allowing data to be passed to it when it's started. These parameters allow runbook actions to be targeted for specific scenarios and environments. This article describes the configuration and use of input parameters in your runbooks.

You can configure input parameters for PowerShell, PowerShell Workflow, graphical, and Python runbooks. A runbook can have multiple parameters with different data types, or no parameters at all. Input parameters can be mandatory or optional, and you can use default values for optional parameters.

You assign values to the input parameters for a runbook when you start it. You can start a runbook from the Azure portal, a web service, or PowerShell. You can also start one as a child runbook that is called inline in another runbook.

Configure input parameters in PowerShell runbooks

PowerShell and PowerShell Workflow runbooks in Azure Automation support input parameters that are defined through the following properties.

PROPERTY	DESCRIPTION
Type	Required. The data type expected for the parameter value. Any .NET type is valid.
Name	Required. The name of the parameter. This name must be unique within the runbook, must start with a letter, and can contain only letters, numbers, or underscore characters.
Mandatory	Optional. Boolean value specifying if the parameter requires a value. If you set this to True, a value must be provided when the runbook is started. If you set this to False, a value is optional. If you don't specify a value for the <code>Mandatory</code> property, PowerShell considers the input parameter optional by default.
Default value	Optional. A value that is used for the parameter if no input value is passed in when the runbook starts. The runbook can set a default value for any parameter.

Windows PowerShell supports more attributes of input parameters than those listed above, such as validation, aliases, and parameter sets. However, Azure Automation currently supports only the listed input parameter properties.

As an example, let's look at a parameter definition in a PowerShell Workflow runbook. This definition has the following general form, where multiple parameters are separated by commas.

```
Param
(
    [Parameter (Mandatory= $true/$false)]
    [Type] $Name1 = <Default value>,

    [Parameter (Mandatory= $true/$false)]
    [Type] $Name2 = <Default value>
)
```

Now let's configure the input parameters for a PowerShell Workflow runbook that outputs details about virtual machines, either a single VM or all VMs within a resource group. This runbook has two parameters, as shown in the following screenshot: the name of the virtual machine (`VMName`) and the name of the resource group (`resourceGroupName`).

```

1 workflow ListAllVMs-PSW
2 {
3     Param(
4         [Parameter(Mandatory = $false)]
5         [String] $VmName,
6         [Parameter(Mandatory = $false)]
7         [String] $ResourceGroupName = "InfraLab"
8     )
9     $connectionName = "AzureRunAsConnection"
10    $subId = Get-AutomationVariable -Name 'SubscriptionId'
11    try
12    {
13        # Get the connection "AzureRunAsConnection"
14        $servicePrincipalConnection=Get-AutomationConnection -Name $connectionName
15
16        "Logging in to Azure..."
17        Add-AzureRmAccount `
18            -ServicePrincipal `
19            -TenantId $servicePrincipalConnection.TenantId `
20            -ApplicationId $servicePrincipalConnection.ApplicationId `
21            -CertificateThumbprint $servicePrincipalConnection.CertificateThumbprint
22        "Setting context to a specific subscription"
23        Set-AzureRmContext -SubscriptionId $subId
24    }
25    catch {
26        if (!$servicePrincipalConnection)
27        {
28            $errorMessage = "Connection $connectionName not found."
29            throw $errorMessage
30        } else{
31            Write-Error -Message $_.Exception
32            throw $_.Exception
33        }
34    }
35    IF ($VmName)
36    {
37        Get-AzureRmVM -Name $VmName -ResourceGroup $ResourceGroupName
38    }
39    else
40    {
41        Get-AzureRmVM -ResourceGroup $ResourceGroupName
42    }
43 }

```

In this parameter definition, the input parameters are simple parameters of type string.

Note that PowerShell and PowerShell Workflow runbooks support all simple types and complex types, such as `[Object]` or `[PSCredential]` for input parameters. If your runbook has an object input parameter, you must use a PowerShell hashtable with name-value pairs to pass in a value. For example, you have the following parameter in a runbook.

```

[Parameter (Mandatory = $true)]
[object] $FullName

```

In this case, you can pass the following value to the parameter.

```

@{ "FirstName"="Joe"; "MiddleName"="Bob"; "LastName"="Smith" }

```

NOTE

When you do not pass a value to an optional String parameter with a null default value, the value of the parameter is an empty string instead of Null.

Configure input parameters in graphical runbooks

To illustrate the configuration of input parameters for a graphical runbook, let's create a runbook that outputs details about virtual machines, either a single VM or all VMs within a resource group. For details, see [My first graphical runbook](#).

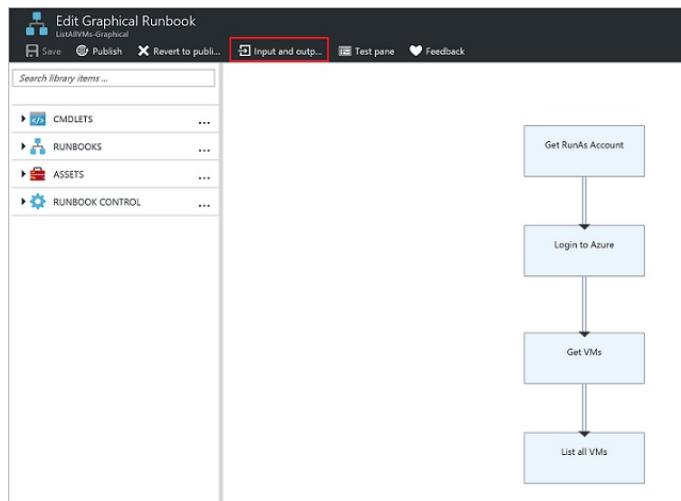
A graphical runbook uses these major runbook activities:

- Configuration of the Azure Run As account to authenticate with Azure.
- Definition of a `Get-AzVM` cmdlet to get VM properties.
- Use of the `Write-Output` activity to output the VM names.

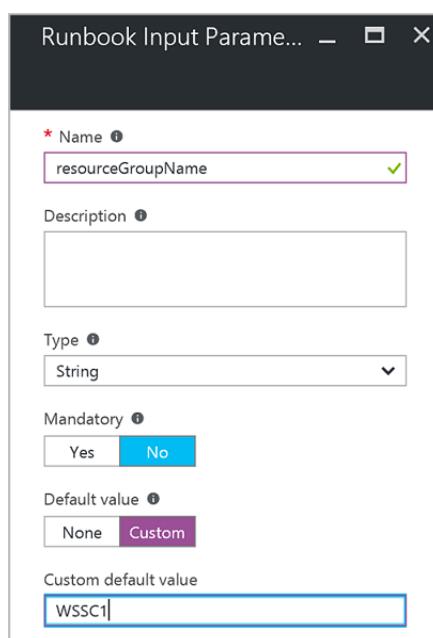
The `Get-AzVM` activity defines two inputs, the VM name and the resource group name. Since these names can be different each time the runbook starts, you must add input parameters to your runbook to accept these inputs. Refer to [Graphical authoring in Azure Automation](#).

Follow these steps to configure the input parameters.

1. Select the graphical runbook from the Runbooks page and then click **Edit**.
2. In the graphical editor, click the **Input and output** button then **Add input** to open the Runbook Input Parameter pane.



3. The Input and Output control displays a list of input parameters that are defined for the runbook. Here you can either add a new input parameter or edit the configuration of an existing input parameter. To add a new parameter for the runbook, click **Add input** to open the **Runbook input parameter** blade, where you can configure parameters using the properties defined in [Graphical authoring in Azure Automation](#).



4. Create two parameters with the following properties to be used by the `Get-AzVM` activity, and then click **OK**.

- Parameter 1:
 - **Name** -- VMName
 - **Type** -- String
 - **Mandatory** -- No
- Parameter 2:
 - **Name** -- resourceGroupName
 - **Type** -- String
 - **Mandatory** -- No
 - **Default value** -- Custom
 - Custom default value -- Name of the resource group that contains the VMs

5. View the parameters in the Input and Output control.

6. Click **OK** again, and then click **Save**.

7. Click **Publish** to publish your runbook.

Configure input parameters in Python runbooks

Unlike PowerShell, PowerShell Workflow, and graphical runbooks, Python runbooks do not take named parameters. The runbook editor parses all input parameters as an array of argument values. You can access the array by importing the `sys` module into your Python script, and then using the `sys.argv` array. It is important to note that the first element of the array, `sys.argv[0]`, is the name of the script. Therefore the first actual input parameter is `sys.argv[1]`.

For an example of how to use input parameters in a Python runbook, see [My first Python runbook in Azure Automation](#).

Assign values to input parameters in runbooks

This section describes several ways to pass values to input parameters in runbooks. You can assign parameter values when you:

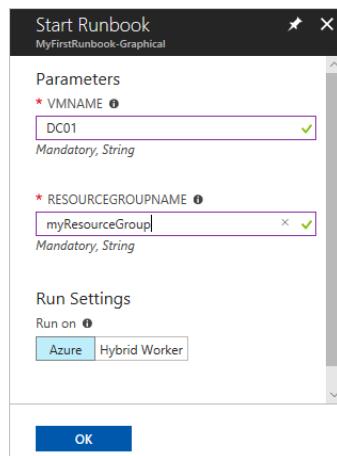
- [Start a runbook](#)
- [Test a runbook](#)
- [Link a schedule for the runbook](#)
- [Create a webhook for the runbook](#)

Start a runbook and assign parameters

A runbook can be started in many ways: through the Azure portal, with a webhook, with PowerShell cmdlets, with the REST API, or with the SDK.

Start a published runbook using the Azure portal and assign parameters

When you [start the runbook](#) in the Azure portal, the **Start Runbook** blade opens and you can enter values for the parameters that you have created.



In the label beneath the input box, you can see the properties that have been set to define parameter attributes, for example, mandatory or optional, type, default value. The help balloon next to the parameter name also defines the key information needed to make decisions about parameter input values.

NOTE

String parameters support empty values of type String. Entering `[EmptyString]` in the input parameter box passes an empty string to the parameter. Also, string parameters don't support Null. If you don't pass any value to a string parameter, PowerShell interprets it as Null.

Start a published runbook using PowerShell cmdlets and assign parameters

- **Azure Resource Manager cmdlets:** You can start an Automation runbook that was created in a resource group by using [Start-AzAutomationRunbook](#).

```
$params = @{"VMName"="WSVMClassic"; "resourceGroupName"="WSVMClassicSG"}  
  
Start-AzAutomationRunbook -AutomationAccountName "TestAutomation" -Name "Get-AzureVMGraphical" -  
ResourceGroupName $resourceGroupName -Parameters $params
```

- **Azure classic deployment model cmdlets:** You can start an automation runbook that was created in a default resource group by using [Start-AzureAutomationRunbook](#).

```
$params = @{"VMName"="WSVMClassic"; "ServiceName"="WSVMClassicSG"}  
  
Start-AzureAutomationRunbook -AutomationAccountName "TestAutomation" -Name "Get-AzureVMGraphical" -  
Parameters $params
```

NOTE

When you start a runbook using PowerShell cmdlets, a default parameter, `MicrosoftApplicationManagementStartedBy`, is created with the value `PowerShell`. You can view this parameter on the Job details pane.

Start a runbook using an SDK and assign parameters

- **Azure Resource Manager method:** You can start a runbook using the SDK of a programming language. Below is a C# code snippet for starting a runbook in your Automation account. You can view all the code at our [GitHub repository](#).

```

public Job StartRunbook(string runbookName, IDictionary<string, string> parameters = null)
{
    var response = AutomationClient.Jobs.Create(resourceGroupName, automationAccount, new
JobCreateParameters
{
    Properties = new JobCreateProperties
    {
        Runbook = new RunbookAssociationProperty
        {
            Name = runbookName
        },
        Parameters = parameters
    }
});
return response.Job;
}

```

- **Azure classic deployment model method:** You can start a runbook by using the SDK of a programming language. Below is a C# code snippet for starting a runbook in your Automation account. You can view all the code at our [GitHub repository](#).

```

public Job StartRunbook(string runbookName, IDictionary<string, string> parameters = null)
{
    var response = AutomationClient.Jobs.Create(automationAccount, new JobCreateParameters
{
    Properties = new JobCreateProperties
    {
        Runbook = new RunbookAssociationProperty
        {
            Name = runbookName
        },
        Parameters = parameters
    }
});
return response.Job;
}

```

To start this method, create a dictionary to store the runbook parameters `VMName` and `resourceGroupName` and their values. Then start the runbook. Below is the C# code snippet for calling the method that's defined above.

```

IDictionary<string, string> RunbookParameters = new Dictionary<string, string>();

// Add parameters to the dictionary.
RunbookParameters.Add("VMName", "WSVMClassic");
RunbookParameters.Add("resourceGroupName", "WSSC1");

//Call the StartRunbook method with parameters
StartRunbook("Get-AzureVMGraphical", RunbookParameters);

```

Start a runbook using the REST API and assign parameters

You can create and start a runbook job with the Azure Automation REST API by using the `PUT` method with the following request URI:

```
https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Automation/automationAccounts/{automationAccountName}/jobs?api-version=2017-05-15-preview
```

In the request URI, replace the following parameters:

- `subscriptionId`: Your Azure subscription ID.
- `resourceGroupName`: The name of the resource group for the Automation account.
- `automationAccountName`: The name of the Automation account that's hosted within the specified cloud service.
- `jobName`: The GUID for the job. GUIDs in PowerShell can be created by using `[GUID]::NewGuid().ToString()*`.

To pass parameters to the runbook job, use the request body. It takes the following information, provided in JSON format:

- Runbook name: Required. The name of the runbook for the job to start.
- Runbook parameters: Optional. A dictionary of the parameter list in (name, value) format, where name is of type String and value can be any valid JSON value.

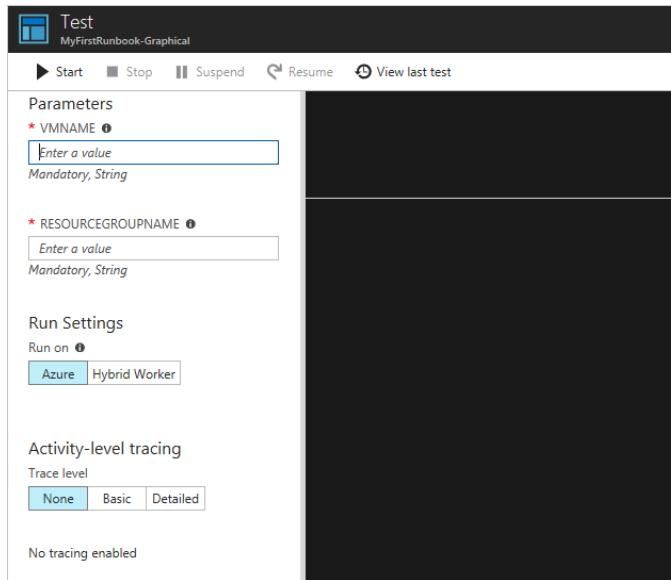
If you want to start the `Get-AzureVMT textual` runbook created earlier with `VMName` and `resourceGroupName` as parameters, use the following JSON format for the request body.

```
{
    "properties": {
        "runbook": {
            "name": "Get-AzureVMT textual"
        },
        "parameters": {
            "VMName": "WindowsVM",
            "resourceGroupName": "ContosoSales"
        }
    }
}
```

An HTTP status code 201 is returned if the job is successfully created. For more information on response headers and the response body, see [create a runbook job by using the REST API](#).

Test a runbook and assign parameters

When you [test the draft version of your runbook](#) by using the test option, the Test page opens. Use this page to configure values for the parameters that you have created.



Link a schedule to a runbook and assign parameters

You can [link a schedule](#) to your runbook so that the runbook starts at a specific time. You assign input parameters when you create the schedule, and the runbook uses these values when it is started by the schedule. You can't save the schedule until all mandatory parameter values are provided.

Create a webhook for a runbook and assign parameters

You can create a [webhook](#) for your runbook and configure runbook input parameters. You can't save the webhook until all mandatory parameter values are provided.

When you execute a runbook by using a webhook, the predefined input parameter

[WebhookData](automation-webhooks.md) is sent, along with the input parameters that you define.

The screenshot shows the Azure portal interface for a graphical runbook. On the left, the runbook details are shown: Job ID, Created, Last Update, Run As, User, and Ran on Azure. The 'Input' tab is selected, displaying three input parameters: VMNAME, RESOURCEGROUPNAME, and WEBHOOKDATA. The WEBHOOKDATA parameter is highlighted with a red border. On the right, the 'Input Parameters' pane shows the values for each parameter. Below the tabs, there are sections for Errors (1), Warnings (0), and Logs.

Pass a JSON object to a runbook

It can be useful to store data that you want to pass to a runbook in a JSON file. For example, you might create a JSON file that contains all parameters that you want to pass to a runbook. To do this, you must convert the JSON code to a string and then convert the string to a PowerShell object before passing it to the runbook.

This section uses an example in which a PowerShell script calls `Start-AzAutomationRunbook` to start a PowerShell runbook, passing the contents of the JSON file to the runbook. The PowerShell runbook starts an Azure VM by retrieving the parameters for the VM from the JSON object.

Create the JSON file

Type the following code in a text file, and save it as `test.json` somewhere on your local computer.

```
{  
    "VmName" : "TestVM",  
    "ResourceGroup" : "AzureAutomationTest"  
}
```

Create the runbook

Create a new PowerShell runbook named `Test-Json` in Azure Automation.

To accept the JSON data, the runbook must take an object as an input parameter. The runbook can then use the properties defined in the JSON file.

```
Param(  
    [parameter(Mandatory=$true)]  
    [object]$json  
)  
  
# Connect to Azure with user-assigned managed identity  
Connect-AzAccount -Identity  
$identity = Get-AzUserAssignedIdentity -ResourceGroupName <ResourceGroupName> -Name <UserAssignedManagedIdentity>  
Connect-AzAccount -Identity -AccountId $identity.ClientId  
  
# Convert object to actual JSON  
$json = $json | ConvertFrom-Json  
  
# Use the values from the JSON object as the parameters for your command  
Start-AzVM -Name $json.VMName -ResourceGroupName $json.ResourceGroup
```

Save and publish this runbook in your Automation account.

Call the runbook from PowerShell

Now you can call the runbook from your local machine by using Azure PowerShell.

1. Sign in to Azure as shown. Afterward, you're prompted to enter your Azure credentials.

```
Connect-AzAccount
```

NOTE

For PowerShell runbooks, `Add-AzAccount` and `Add-AzureRMAccount` are aliases for `Connect-AzAccount`. Note that these aliases are not available for graphical runbooks. A graphical runbook can only use `Connect-AzAccount` itself.

2. Get the contents of the saved JSON file and convert it to a string. `JsonPath` indicates the path where you saved the JSON file.

```
$json = (Get-content -path 'JsonPath\test.json' -Raw) | Out-string
```

3. Convert the string contents of `$json` to a PowerShell object.

```
$JsonParams = @{"json"=$json}
```

4. Create a hashtable for the parameters for `Start-AzAutomationRunbook`.

```
$RBPParams = @{
    AutomationAccountName = 'AATest'
    ResourceGroupName = 'RGTest'
    Name = 'Test-Json'
    Parameters = $JsonParams
}
```

Notice that you're setting the value of `Parameters` to the PowerShell object that contains the values from the JSON file.

5. Start the runbook.

```
$job = Start-AzAutomationRunbook @RBPParams
```

Next steps

- To prepare a textual runbook, see [Edit textual runbooks in Azure Automation](#).
- To prepare a graphical runbook, see [Author graphical runbooks in Azure Automation](#).

Manage runbooks in Azure Automation

5/4/2021 • 12 minutes to read • [Edit Online](#)

You can add a runbook to Azure Automation by either creating a new one or importing an existing one from a file or the [Runbook Gallery](#). This article provides information for managing a runbook imported from a file. You can find all the details of accessing community runbooks and modules in [Runbook and module galleries for Azure Automation](#).

Create a runbook

Create a new runbook in Azure Automation using the Azure portal or Windows PowerShell. Once the runbook has been created, you can edit it using information in:

- [Edit textual runbook in Azure Automation](#)
- [Learn key Windows PowerShell Workflow concepts for Automation runbooks](#)
- [Manage Python 2 packages in Azure Automation](#)
- [Manage Python 3 packages \(preview\) in Azure Automation](#)

Create a runbook in the Azure portal

1. Sign in to the Azure [portal](#).
2. Search for and select **Automation Accounts**.
3. On the **Automation Accounts** page, select your Automation account from the list.
4. From the Automation account, select **Runbooks** under **Process Automation** to open the list of runbooks.
5. Click **Create a runbook**.
6. Enter a name for the runbook and select its [type](#). The runbook name must start with a letter and can contain letters, numbers, underscores, and dashes.
7. Click **Create** to create the runbook and open the editor.

Create a runbook with PowerShell

Use the `New-AzAutomationRunbook` cmdlet to create an empty runbook. Use the `Type` parameter to specify one of the runbook types defined for `New-AzAutomationRunbook`.

The following example shows how to create a new empty runbook.

```
$params = @{
    AutomationAccountName = 'MyAutomationAccount'
    Name                  = 'NewRunbook'
    ResourceGroupName     = 'MyResourceGroup'
    Type                 = 'PowerShell'
}
New-AzAutomationRunbook @params
```

Import a runbook

You can import a PowerShell or PowerShell Workflow (.ps1) script, a graphical runbook (.graphrunbook), or a Python 2 or Python 3 script (.py) to make your own runbook. You specify the [type of runbook](#) that is created during import, taking into account the following considerations.

- You can import a .ps1 file that doesn't contain a workflow into either a [PowerShell runbook](#) or a [PowerShell Workflow runbook](#). If you import it into a PowerShell Workflow runbook, it is converted to a

workflow. In this case, comments are included in the runbook to describe the changes made.

- You can import only a **.ps1** file containing a PowerShell Workflow into a [PowerShell Workflow runbook](#). If the file contains multiple PowerShell workflows, the import fails. You have to save each workflow to its own file and import each separately.
- Do not import a **.ps1** file containing a PowerShell Workflow into a [PowerShell runbook](#), as the PowerShell script engine can't recognize it.
- Only import a **.graphrunbook** file into a new [graphical runbook](#).

Import a runbook from the Azure portal

You can use the following procedure to import a script file into Azure Automation.

NOTE

You can only import a **.ps1** file into a PowerShell Workflow runbook using the portal.

1. In the Azure portal, search for and select **Automation Accounts**.
2. On the **Automation Accounts** page, select your Automation account from the list.
3. From the Automation account, select **Runbooks** under **Process Automation** to open the list of runbooks.
4. Click **Import a runbook**.
5. Click **Runbook file** and select the file to import.
6. If the **Name** field is enabled, you have the option of changing the runbook name. The name must start with a letter and can contain letters, numbers, underscores, and dashes.
7. The **runbook type** is automatically selected, but you can change the type after taking the applicable restrictions into account.
8. Click **Create**. The new runbook appears in the list of runbooks for the Automation account.
9. You have to [publish the runbook](#) before you can run it.

NOTE

After you import a graphical runbook, you can convert it to another type. However, you can't convert a graphical runbook to a textual runbook.

Import a runbook with Windows PowerShell

Use the [Import-AzAutomationRunbook](#) cmdlet to import a script file as a draft runbook. If the runbook already exists, the import fails unless you use the **Force** parameter with the cmdlet.

The following example shows how to import a script file into a runbook.

```
$params = @{
    AutomationAccountName = 'MyAutomationAccount'
    Name                  = 'Sample_TestRunbook'
    ResourceGroupName     = 'MyResourceGroup'
    Type                 = 'PowerShell'
    Path                 = 'C:\Runbooks\Sample_TestRunbook.ps1'
}
Import-AzAutomationRunbook @params
```

Handle resources

If your runbook creates a [resource](#), the script should check to see if the resource already exists before attempting to create it. Here's a basic example.

```

$vmName = 'WindowsVM1'
$rgName = 'MyResourceGroup'
$myCred = Get-AutomationPSCredential 'MyCredential'

$vmExists = Get-AzResource -Name $vmName -ResourceGroupName $rgName
if (-not $vmExists) {
    Write-Output "VM $vmName does not exist, creating"
    New-AzVM -Name $vmName -ResourceGroupName $rgName -Credential $myCred
} else {
    Write-Output "VM $vmName already exists, skipping"
}

```

Retrieve details from Activity log

You can retrieve runbook details, such as the person or account that started a runbook, from the [Activity log](#) for the Automation account. The following PowerShell example provides the last user to run the specified runbook.

```

$rgName = 'MyResourceGroup'
$accountName = 'MyAutomationAccount'
$runbookName = 'MyRunbook'
$startTime = (Get-Date).AddDays(-1)

$params = @{
    ResourceGroupName = $rgName
    StartTime        = $startTime
}
$JobActivityLogs = (Get-AzLog @params).Where( { $_.Authorization.Action -eq
'Microsoft.Automation/automationAccounts/jobs/write' })

$JobInfo = @{}
foreach ($log in $JobActivityLogs) {
    # Get job resource
    $JobResource = Get-AzResource -ResourceId $log.ResourceId

    if ($null -eq $JobInfo[$log.SubmissionTimestamp] -and $JobResource.Properties.Runbook.Name -eq
$runbookName) {
        # Get runbook
        $jobParams = @{
            ResourceGroupName      = $rgName
            AutomationAccountName = $accountName
            Id                   = $JobResource.Properties.JobId
        }
        $Runbook = Get-AzAutomationJob @jobParams | Where-Object RunbookName -EQ $runbookName

        # Add job information to hashtable
        $JobInfo.Add($log.SubmissionTimestamp, @{$Runbook.RunbookName, $Log.Caller,
$JobResource.Properties.jobId})
    }
}
$JobInfo.GetEnumerator() | Sort-Object Key -Descending | Select-Object -First 1

```

Track progress

It's a good practice to author your runbooks to be modular in nature, with logic that can be reused and restarted easily. Tracking progress in a runbook ensures that the runbook logic executes correctly if there are issues.

You can track the progress of a runbook by using an external source, such as a storage account, a database, or shared files. Create logic in your runbook to first check the state of the last action taken. Then, based on the results of the check, the logic can either skip or continue specific tasks in the runbook.

Prevent concurrent jobs

Some runbooks behave strangely if they run across multiple jobs at the same time. In this case, it's important for a runbook to implement logic to determine if there is already a running job. Here's a basic example.

```
# Authenticate to Azure
$connection = Get-AutomationConnection -Name AzureRunAsConnection
$cnParams = @{
    ServicePrincipal      = $true
    Tenant                = $connection.TenantId
    ApplicationId         = $connection.ApplicationId
    CertificateThumbprint = $connection.CertificateThumbprint
}
Connect-AzAccount @cnParams
$AzureContext = Set-AzContext -SubscriptionId $connection.SubscriptionID

# Check for already running or new runbooks
$runbookName = "RunbookName"
$rgName = "ResourceGroupName"
$accountName = "AutomationAccountName"
$jobs = Get-AzAutomationJob -ResourceGroupName $rgName -AutomationAccountName $accountName -RunbookName
$runbookName -AzContext $AzureContext

# Check to see if it is already running
$runningCount = ($jobs.Where( { $_.Status -eq 'Running' })).count

if (($jobs.Status -contains 'Running' -and $runningCount -gt 1 ) -or ($jobs.Status -eq 'New')) {
    # Exit code
    Write-Output "Runbook [$runbookName] is already running"
    exit 1
} else {
    # Insert Your code here
}
```

Alternatively, you can use PowerShell's splatting feature to pass the connection information to `Connect-AzAccount`. In that case, the first few lines of the previous sample would look like this.

```
# Authenticate to Azure
$connection = Get-AutomationConnection -Name AzureRunAsConnection
Connect-AzAccount @connection
$AzureContext = Set-AzContext -SubscriptionId $connection.SubscriptionID
```

For more information, see [about splatting](#).

Handle transient errors in a time-dependent script

Your runbooks must be robust and capable of handling [errors](#), including transient errors that can cause them to restart or fail. If a runbook fails, Azure Automation retries it.

If your runbook normally runs within a time constraint, have the script implement logic to check the execution time. This check ensures the running of operations such as startup, shutdown, or scale-out only during specific times.

NOTE

The local time on the Azure sandbox process is set to UTC. Calculations for date and time in your runbooks must take this fact into consideration.

Work with multiple subscriptions

Your runbook must be able to work with [subscriptions](#). For example, to handle multiple subscriptions, the runbook uses the [Disable-AzContextAutosave](#) cmdlet. This cmdlet ensures that the authentication context isn't retrieved from another runbook running in the same sandbox.

```
Disable-AzContextAutosave -Scope Process

$connection = Get-AutomationConnection -Name AzureRunAsConnection
$cnParams = @{
    ServicePrincipal      = $true
    Tenant                = $connection.TenantId
    ApplicationId         = $connection.ApplicationId
    CertificateThumbprint = $connection.CertificateThumbprint
}
Connect-AzAccount @cnParams

$childRunbookName = 'ChildRunbookDemo'
$accountName = 'MyAutomationAccount'
$rgName = 'MyResourceGroup'

$startParams = @{
    ResourceGroupName     = $rgName
    AutomationAccountName = $accountName
    Name                 = $childRunbookName
    DefaultProfile       = $AzureContext
}
Start-AzAutomationRunbook @startParams
```

Work with a custom script

NOTE

You can't normally run custom scripts and runbooks on the host with a Log Analytics agent installed.

To use a custom script:

1. Create an Automation account.
2. Deploy the [Hybrid Runbook Worker](#) role.
3. If on a Linux machine, you need elevated privileges. Sign in to [turn off signature checks](#).

Test a runbook

When you test a runbook, the [Draft version](#) is executed and any actions that it performs are completed. No job history is created, but the [output](#) and [warning and error](#) streams are displayed in the [Test output](#) pane.

Messages to the [verbose stream](#) are displayed in the Output pane only if the [VerbosePreference](#) variable is set to [Continue](#).

Even though the Draft version is being run, the runbook still executes normally and performs any actions against resources in the environment. For this reason, you should only test runbooks on non-production resources.

The procedure to test each [type of runbook](#) is the same. There's no difference in testing between the textual editor and the graphical editor in the Azure portal.

1. Open the Draft version of the runbook in either the [textual editor](#) or the [graphical editor](#).
2. Click **Test** to open the **Test** page.

3. If the runbook has parameters, they're listed in the left pane, where you can provide values to be used for the test.
4. If you want to run the test on a [Hybrid Runbook Worker](#), change **Run Settings** to **Hybrid Worker** and select the name of the target group. Otherwise, keep the default **Azure** to run the test in the cloud.
5. Click **Start** to begin the test.
6. You can use the buttons under the **Output** pane to stop or suspend a [PowerShell Workflow](#) or graphical runbook while it's being tested. When you suspend the runbook, it completes the current activity before being suspended. Once the runbook is suspended, you can stop it or restart it.
7. Inspect the output from the runbook in the **Output** pane.

Publish a runbook

When you create or import a new runbook, you have to publish it before you can run it. Each runbook in Azure Automation has a Draft version and a Published version. Only the Published version is available to be run, and only the Draft version can be edited. The Published version is unaffected by any changes to the Draft version. When the Draft version should be made available, you publish it, overwriting the current Published version with the Draft version.

Publish a runbook in the Azure portal

1. From the Azure portal, open the runbook in your Automation account.
2. Click **Edit**.
3. Click **Publish** and then **Yes** in response to the verification message.

Publish a runbook using PowerShell

Use the [Publish-AzAutomationRunbook](#) cmdlet to publish your runbook.

```
$accountName = "MyAutomationAccount"
$runbookName = "Sample_TestRunbook"
$rgName = "MyResourceGroup"

$publishParams = @{
    AutomationAccountName = $accountName
    ResourceGroupName      = $rgName
    Name                  = $runbookName
}
Publish-AzAutomationRunbook @publishParams
```

Schedule a runbook in the Azure portal

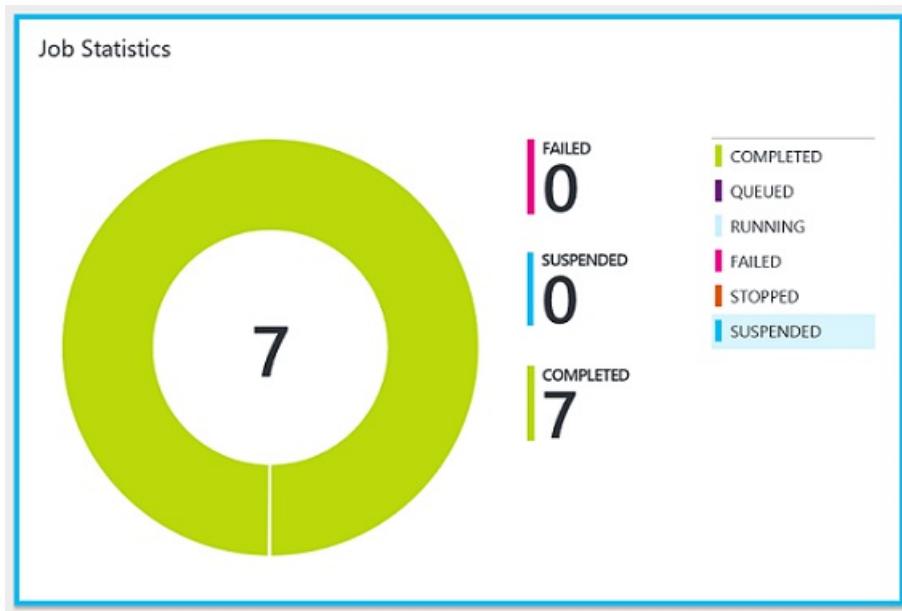
When your runbook has been published, you can schedule it for operation:

1. From the Azure portal, open the runbook in your Automation account.
2. Select **Schedules** under **Resources**.
3. Select **Add a schedule**.
4. In the Schedule Runbook pane, select **Link a schedule to your runbook**.
5. Choose **Create a new schedule** in the Schedule pane.
6. Enter a name, description, and other parameters in the New schedule pane.
7. Once the schedule is created, highlight it and click **OK**. It should now be linked to your runbook.
8. Look for an email in your mailbox to notify you of the runbook status.

Obtain job statuses

View statuses in the Azure portal

Details of job handling in Azure Automation are provided in [Jobs](#). When you are ready to see your runbook jobs, use Azure portal and access your Automation account. On the right, you can see a summary of all the runbook jobs in **Job Statistics**.



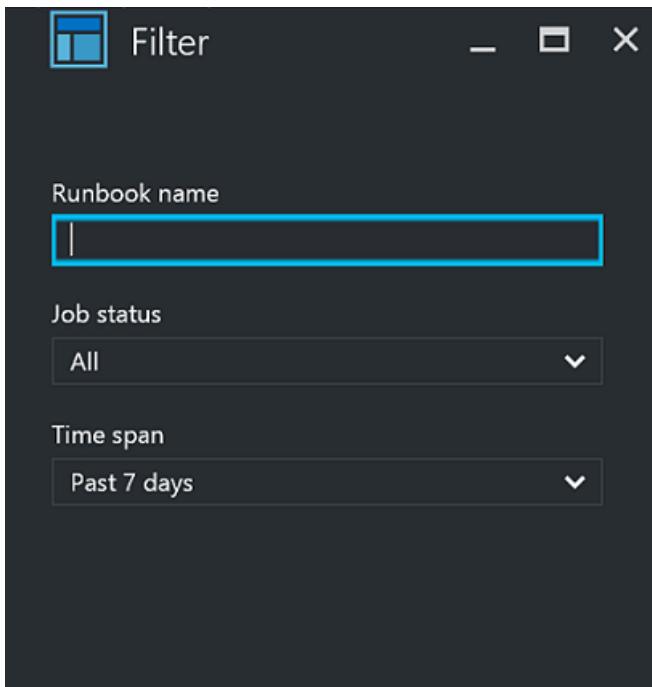
The summary displays a count and graphical representation of the job status for each job executed.

Clicking the tile presents the **Jobs** page, which includes a summarized list of all jobs executed. This page shows the status, runbook name, start time, and completion time for each job.

The screenshot shows the 'Jobs' page with a dark header containing a 'Jobs' icon and a 'Filter Jobs' button. The main area is a table with four columns: STATUS, RUNBOOK, CREATED, and LAST UPDATED. All listed jobs are 'Completed' with a green checkmark icon. The table has 8 rows, each representing a completed runbook named 'Stop-AllVMs' at different dates and times.

STATUS	RUNBOOK	CREATED	LAST UPDATED
✓ Completed	Stop-AllVMs	11/1/2016 6:00 PM	11/1/2016 6:08 PM
✓ Completed	Stop-AllVMs	10/31/2016 6:00 PM	10/31/2016 6:12 PM
✓ Completed	Stop-AllVMs	10/30/2016 6:00 PM	10/30/2016 6:08 PM
✓ Completed	Stop-AllVMs	10/29/2016 6:00 PM	10/29/2016 6:09 PM
✓ Completed	Stop-AllVMs	10/28/2016 6:00 PM	10/28/2016 6:08 PM
✓ Completed	Stop-AllVMs	10/27/2016 6:00 PM	10/27/2016 6:08 PM
✓ Completed	Stop-AllVMs	10/26/2016 6:00 PM	10/26/2016 6:12 PM

You can filter the list of jobs by selecting **Filter jobs**. Filter on a specific runbook, job status, or a choice from the dropdown list, and provide the time range for the search.



Alternatively, you can view job summary details for a specific runbook by selecting that runbook from the Runbooks page in your Automation account and then selecting **Jobs**. This action presents the Jobs page. From here, you can click a job record to view its details and output.

The screenshot shows a job summary for a runbook named "Stop-AllVMs" from 11/1/2016 at 6:00 PM. The job is completed with 1 error and 0 warnings. The status bar indicates 0 inputs and 0 outputs. The job ID is edf36252-6cc9-4c59-92d7-dcaeb2285cac. The runbook was created and last updated on 11/1/2016 at 6:08 PM. The job ran on Azure.

Errors	Warnings	All Logs
1 ✗	0 !	(0)

Retrieve job statuses using PowerShell

Use the [Get-AzAutomationJob](#) cmdlet to retrieve the jobs created for a runbook and the details of a particular job. If you start a runbook using [Start-AzAutomationRunbook](#), it returns the resulting job. Use [Get-AzAutomationJobOutput](#) to retrieve job output.

The following example gets the last job for a sample runbook and displays its status, the values provided for the runbook parameters, and the job output.

```

$getJobParams = @{
    AutomationAccountName = 'MyAutomationAccount'
    ResourceGroupName     = 'MyResourceGroup'
    Runbookname           = 'Test-Runbook'
}
$job = (Get-AzAutomationJob @getJobParams | Sort-Object LastModifiedDate -Desc)[0]
$job | Select-Object JobId, Status, JobParameters

$getOutputParams = @{
    AutomationAccountName = 'MyAutomationAccount'
    ResourceGroupName     = 'MyResourceGroup'
    Id                   = $job.JobId
    Stream               = 'Output'
}
Get-AzAutomationJobOutput @getOutputParams

```

The following example retrieves the output for a specific job and returns each record. If there's an [exception](#) for one of the records, the script writes the exception instead of the value. This behavior is useful since exceptions can provide additional information that might not be logged normally during output.

```

$params = @{
    AutomationAccountName = 'MyAutomationAccount'
    ResourceGroupName     = 'MyResourceGroup'
    Stream               = 'Any'
}
$output = Get-AzAutomationJobOutput @params

foreach ($item in $output) {
    $jobOutParams = @{
        AutomationAccountName = 'MyAutomationAccount'
        ResourceGroupName     = 'MyResourceGroup'
        Id                   = $item.StreamRecordId
    }
    $fullRecord = Get-AzAutomationJobOutputRecord @jobOutParams

    if ($fullRecord.Type -eq 'Error') {
        $fullRecord.Value.Exception
    } else {
        $fullRecord.Value
    }
}

```

Next steps

- To learn details of runbook management, see [Runbook execution in Azure Automation](#).
- To prepare a PowerShell runbook, see [Edit textual runbooks in Azure Automation](#).
- To troubleshoot issues with runbook execution, see [Troubleshoot runbook issues](#).

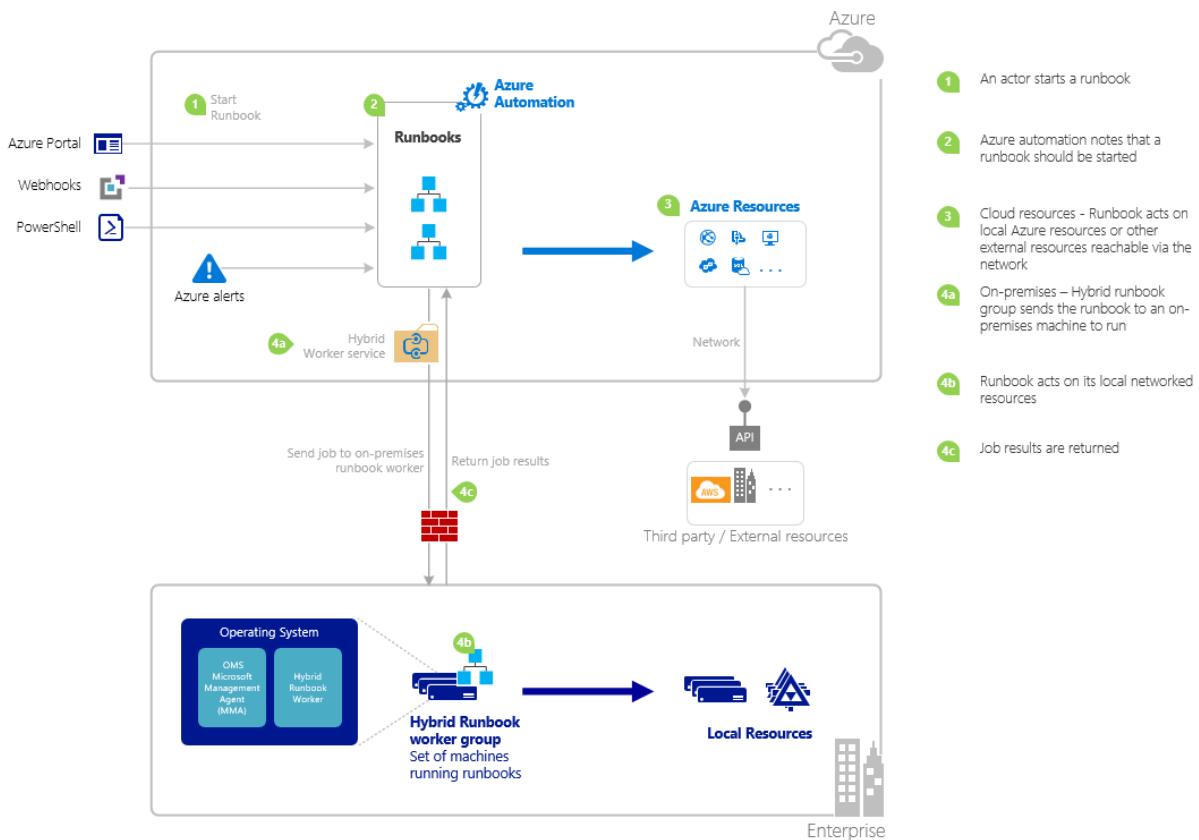
Start a runbook in Azure Automation

4/29/2021 • 5 minutes to read • [Edit Online](#)

The following table helps you determine the method to start a runbook in Azure Automation that is most suitable to your particular scenario. This article includes details on starting a runbook with the Azure portal and with Windows PowerShell. Details on the other methods are provided in other documentation that you can access from the links below.

METHOD	CHARACTERISTICS
Azure portal	<ul style="list-style-type: none">• Simplest method with interactive user interface.• Form to provide simple parameter values.• Easily track job state.• Access authenticated with Azure sign in.
Windows PowerShell	<ul style="list-style-type: none">• Call from command line with Windows PowerShell cmdlets.• Can be included in automated feature with multiple steps.• Request is authenticated with certificate or OAuth user principal / service principal.• Provide simple and complex parameter values.• Track job state.• Client required to support PowerShell cmdlets.
Azure Automation API	<ul style="list-style-type: none">• Most flexible method but also most complex.• Call from any custom code that can make HTTP requests.• Request authenticated with certificate, or OAuth user principal / service principal.• Provide simple and complex parameter values. <i>If you're calling a Python runbook using the API, the JSON payload must be serialized.</i>• Track job state.
Webhooks	<ul style="list-style-type: none">• Start runbook from single HTTP request.• Authenticated with security token in URL.• Client can't override parameter values specified when webhook created. Runbook can define single parameter that is populated with the HTTP request details.• No ability to track job state through webhook URL.
Respond to Azure Alert	<ul style="list-style-type: none">• Start a runbook in response to Azure alert.• Configure webhook for runbook and link to alert.• Authenticated with security token in URL.
Schedule	<ul style="list-style-type: none">• Automatically start runbook on hourly, daily, weekly, or monthly schedule.• Manipulate schedule through Azure portal, PowerShell cmdlets, or Azure API.• Provide parameter values to be used with schedule.
From Another Runbook	<ul style="list-style-type: none">• Use a runbook as an activity in another runbook.• Useful for functionality used by multiple runbooks.• Provide parameter values to child runbook and use output in parent runbook.

The following image illustrates detailed step-by-step process in the life cycle of a runbook. It includes different ways a runbook starts in Azure Automation, which components required for Hybrid Runbook Worker to execute Azure Automation runbooks, and interactions between different components. To learn about executing Automation runbooks in your datacenter, refer to [hybrid runbook workers](#)



Work with runbook parameters

When you start a runbook from the Azure portal or Windows PowerShell, the instruction is sent through the Azure Automation web service. This service doesn't support parameters with complex data types. If you need to provide a value for a complex parameter, then you must call it inline from another runbook as described in [Child runbooks in Azure Automation](#).

The Azure Automation web service provides special functionality for parameters using certain data types as described in the following sections.

Named values

If the parameter is data type [object], then you can use the following JSON format to send it a list of named values: `{Name1:'Value1', Name2:'Value2', Name3:'Value3'}`. These values must be simple types. The runbook receives the parameter as a `PSCustomObject` with properties that correspond to each named value.

Consider the following test runbook that accepts a parameter called user.

```

Workflow Test-Parameters
{
    param (
        [Parameter(Mandatory=$true)][object]$user
    )
    $userObject = $user | ConvertFrom-JSON
    if ($userObject.Show) {
        foreach ($i in 1..$userObject.RepeatCount) {
            $userObject.FirstName
            $userObject.LastName
        }
    }
}

```

The following text could be used for the user parameter.

```
{FirstName:'Joe',LastName:'Smith',RepeatCount:'2',Show:'True'}
```

This results in the following output:

```

Joe
Smith
Joe
Smith

```

Arrays

If the parameter is an array such as [array] or [string[]], then you can use the following JSON format to send it a list of values: *[Value1, Value2, Value3]*. These values must be simple types.

Consider the following test runbook that accepts a parameter called *user*.

```

Workflow Test-Parameters
{
    param (
        [Parameter(Mandatory=$true)][array]$user
    )
    if ($user[3]) {
        foreach ($i in 1..$user[2]) {
            $ user[0]
            $ user[1]
        }
    }
}

```

The following text could be used for the user parameter.

```
["Joe","Smith",2,true]
```

This results in the following output:

```

Joe
Smith
Joe
Smith

```

Credentials

If the parameter is data type `PSCredential`, you can provide the name of an Azure Automation [credential asset](#).

The runbook retrieves the credential with the name that you specify. The following test runbook accepts a parameter called `credential`.

```
Workflow Test-Parameters
{
    param (
        [Parameter(Mandatory=$true)][PSCredential]$credential
    )
    $credential.UserName
}
```

The following text could be used for the user parameter assuming that there was a credential asset called `My Credential`.

```
My Credential
```

Assuming that the user name in the credential is `jsmith`, the following output is displayed.

```
jsmith
```

Start a runbook with the Azure portal

1. In the Azure portal, select **Automation** and then select the name of an Automation account.
2. From the left-hand pane, select **Runbooks**.
3. On the **Runbooks** page, select a runbook, and then click **Start**.
4. If the runbook has parameters, you're prompted to provide values with a text box for each parameter. For more information on parameters, see [Runbook Parameters](#).
5. On the **Job** pane, you can view the status of the runbook job.

Start a runbook with PowerShell

You can use the [Start-AzAutomationRunbook](#) to start a runbook with Windows PowerShell. The following sample code starts a runbook called **Test-Runbook**.

```
Start-AzAutomationRunbook -AutomationAccountName "MyAutomationAccount" -Name "Test-Runbook" -
ResourceGroupName "ResourceGroup01"
```

`Start-AzAutomationRunbook` returns a job object that you can use to track status once the runbook is started. You can then use this job object with [Get-AzAutomationJob](#) to determine the status of the job and [Get-AzAutomationJobOutput](#) to retrieve its output. The following example starts a runbook called **Test-Runbook**, waits until it has completed, and then displays its output.

```

$runbookName = "Test-Runbook"
$ResourceGroup = "ResourceGroup01"
$AutomationAcct = "MyAutomationAccount"

$job = Start-AzAutomationRunbook -AutomationAccountName $AutomationAcct -Name $runbookName -
ResourceGroupName $ResourceGroup

$doLoop = $true
While ($doLoop) {
    $job = Get-AzAutomationJob -AutomationAccountName $AutomationAcct -Id $job.JobId -ResourceGroupName
$ResourceGroup
    $status = $job.Status
    $doLoop = (($status -ne "Completed") -and ($status -ne "Failed") -and ($status -ne "Suspended") -and
($status -ne "Stopped"))
}

Get-AzAutomationJobOutput -AutomationAccountName $AutomationAcct -Id $job.JobId -ResourceGroupName
$ResourceGroup -Stream Output

```

If the runbook requires parameters, then you must provide them as a [hashtable](#). The key of the hashtable must match the parameter name and the value is the parameter value. The following example shows how to start a runbook with two string parameters named FirstName and LastName, an integer named RepeatCount, and a boolean parameter named Show. For more information on parameters, see [Runbook Parameters](#).

```

$params = @{"FirstName"="Joe";"LastName"="Smith";"RepeatCount"=2;"Show"=$true}
Start-AzAutomationRunbook -AutomationAccountName "MyAutomationAccount" -Name "Test-Runbook" -
ResourceGroupName "ResourceGroup01" -Parameters $params

```

Next steps

- For details of runbook management, see [Manage runbooks in Azure Automation](#).
- For PowerShell details, see [PowerShell Docs](#).
- To troubleshoot issues with runbook execution, see [Troubleshoot runbook issues](#).

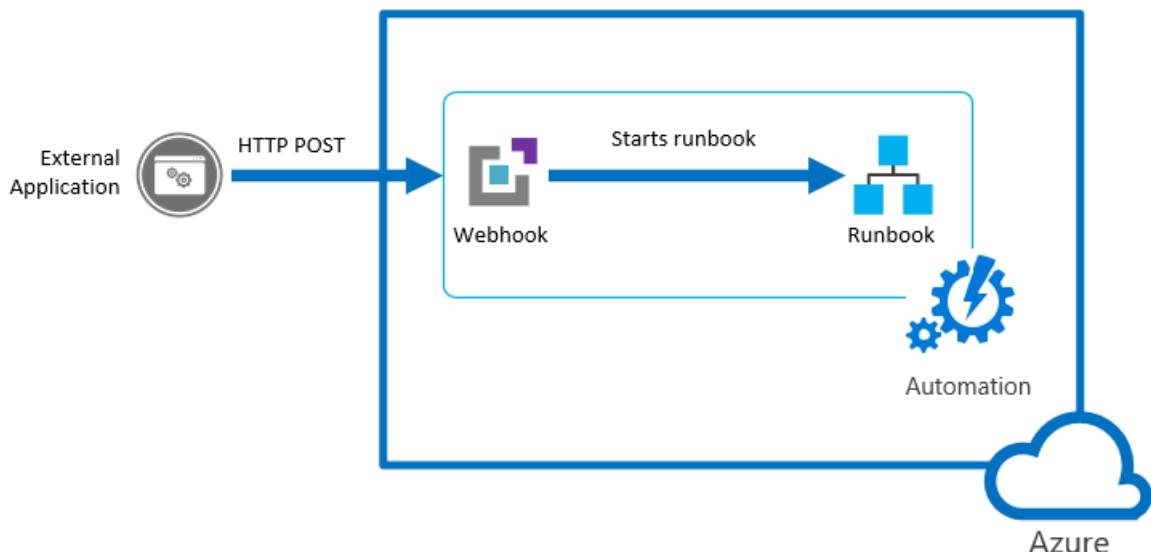
Start a runbook from a webhook

9/10/2021 • 13 minutes to read • [Edit Online](#)

A webhook allows an external service to start a particular runbook in Azure Automation through a single HTTP request. External services include Azure DevOps Services, GitHub, Azure Monitor logs, and custom applications. Such a service can use a webhook to start a runbook without implementing the full Azure Automation API. You can compare webhooks to other methods of starting a runbook in [Starting a runbook in Azure Automation](#).

NOTE

Using a webhook to start a Python runbook is not supported.



To understand client requirements for TLS 1.2 with webhooks, see [TLS 1.2 for Azure Automation](#).

Webhook properties

The following table describes the properties that you must configure for a webhook.

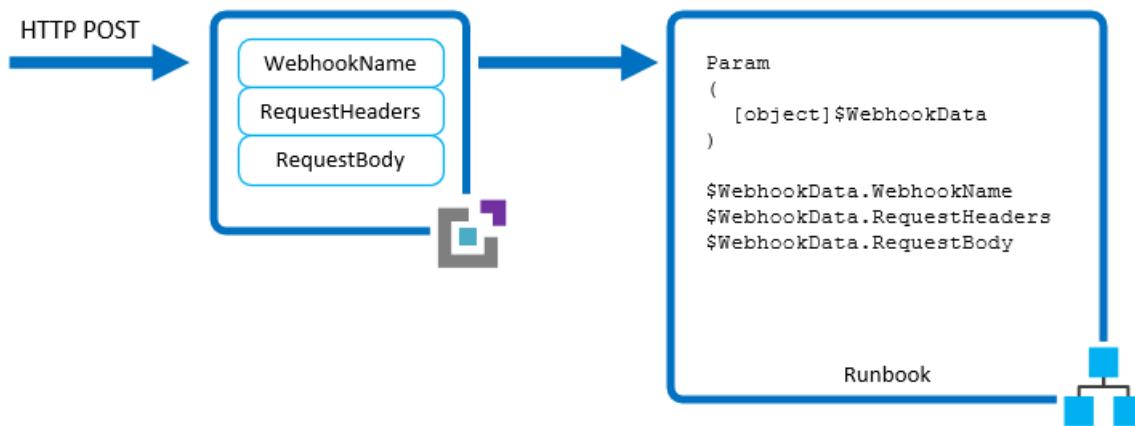
PROPERTY	DESCRIPTION
Name	Name of the webhook. You can provide any name you want, since it isn't exposed to the client. It's only used for you to identify the runbook in Azure Automation. As a best practice, you should give the webhook a name related to the client that uses it.

PROPERTY	DESCRIPTION
URL	<p>URL of the webhook. This is the unique address that a client calls with an HTTP POST to start the runbook linked to the webhook. It's automatically generated when you create the webhook. You can't specify a custom URL.</p> <p>The URL contains a security token that allows a third-party system to invoke the runbook with no further authentication. For this reason, you should treat the URL like a password. For security reasons, you can only view the URL in the Azure portal when creating the webhook. Note the URL in a secure location for future use.</p>
Expiration date	Expiration date of the webhook, after which it can no longer be used. You can modify the expiration date after the webhook is created, as long as the webhook hasn't expired.
Enabled	Setting indicating if the webhook is enabled by default when it's created. If you set this property to Disabled, no client can use the webhook. You can set this property when you create the webhook or any other time after its creation.

Parameters used when the webhook starts a runbook

A webhook can define values for runbook parameters that are used when the runbook starts. The webhook must include values for any mandatory runbook parameters and can include values for optional parameters. A parameter value configured to a webhook can be modified even after webhook creation. Multiple webhooks linked to a single runbook can each use different runbook parameter values. When a client starts a runbook using a webhook, it can't override the parameter values defined in the webhook.

To receive data from the client, the runbook supports a single parameter called `WebhookData`. This parameter defines an object containing data that the client includes in a POST request.



The `WebhookData` parameter has the following properties:

PROPERTY	DESCRIPTION
WebhookName	Name of the webhook.

PROPERTY	DESCRIPTION
RequestHeader	Hashtable containing the headers of the incoming POST request.
RequestBody	Body of the incoming POST request. This body keeps any data formatting, such as string, JSON, XML, or form-encoded. The runbook must be written to work with the data format that is expected.

There's no configuration of the webhook required to support the `WebhookData` parameter, and the runbook isn't required to accept it. If the runbook doesn't define the parameter, any details of the request sent from the client are ignored.

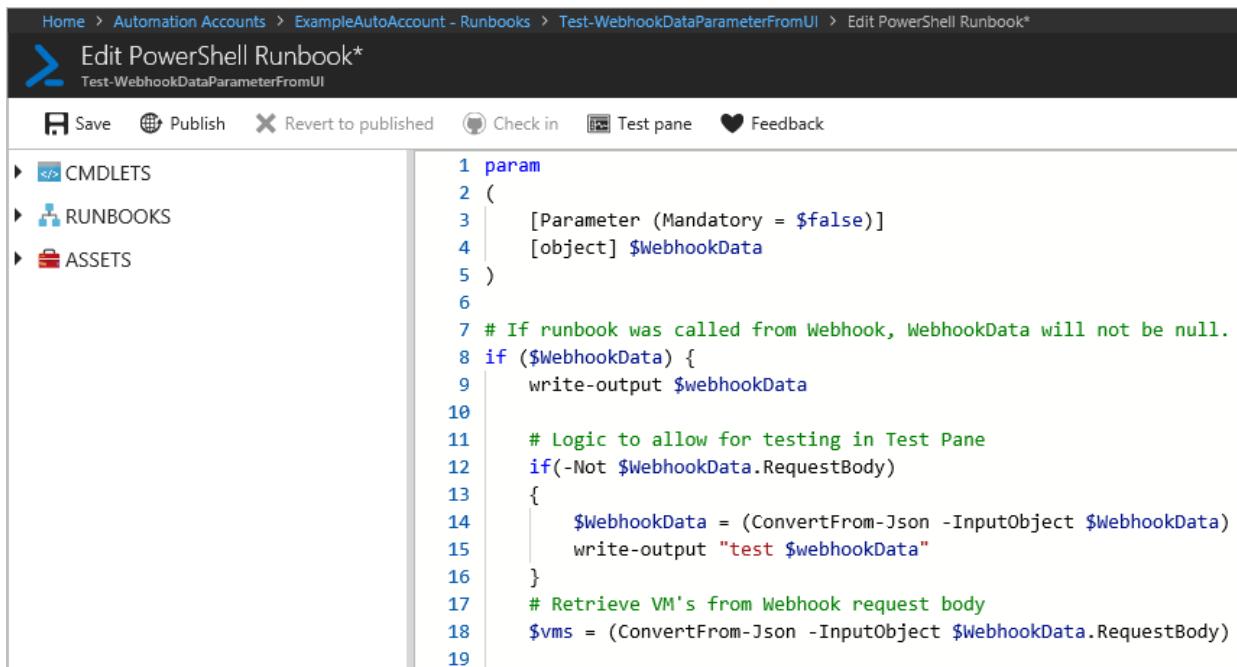
NOTE

When calling a webhook, the client should always store any parameter values in case the call fails. If there is a network outage or connection issue, the application can't retrieve failed webhook calls.

If you specify a value for `WebhookData` at webhook creation, it's overridden when the webhook starts the runbook with the data from the client POST request. This happens even if the application doesn't include any data in the request body.

If you start a runbook that defines `WebhookData` using a mechanism other than a webhook, you can provide a value for `WebhookData` that the runbook recognizes. This value should be an object with the same [properties](#) as the `WebhookData` parameter so that the runbook can work with it just as it works with actual `WebhookData` objects passed by a webhook.

For example, if you're starting the following runbook from the Azure portal and want to pass some sample webhook data for testing, you must pass the data in JSON in the user interface.



```

Home > Automation Accounts > ExampleAutoAccount - Runbooks > Test-WebhookDataParameterFromUI > Edit PowerShell Runbook*
Edit PowerShell Runbook*
Test-WebhookDataParameterFromUI
Save Publish Revert to published Check in Test pane Feedback
▶ CMDLETS
▶ RUNBOOKS
▶ ASSETS
1 param
2 (
3     [Parameter (Mandatory = $false)]
4     [object] $WebhookData
5 )
6
7 # If runbook was called from Webhook, WebhookData will not be null.
8 if ($WebhookData) {
9     write-output $WebhookData
10
11    # Logic to allow for testing in Test Pane
12    if(-Not $WebhookData.RequestBody)
13    {
14        $WebhookData = (ConvertFrom-Json -InputObject $WebhookData)
15        write-output "test $WebhookData"
16    }
17    # Retrieve VM's from Webhook request body
18    $vms = (ConvertFrom-Json -InputObject $WebhookData.RequestBody)
19

```

For the next runbook example, let's define the following properties for `WebhookData`:

- **WebhookName:** MyWebhook
- **RequestBody:**

```
*[{'ResourceGroup': 'myResourceGroup', 'Name': 'vm01'}, {'ResourceGroup': 'myResourceGroup', 'Name': 'vm02'}]*
```

Now we pass the following JSON object in the UI for the `WebhookData` parameter. This example, with carriage returns and newline characters, matches the format that is passed in from a webhook.

```
{"WebhookName": "mywebhook", "RequestBody": "[\r\n {\r\n \"ResourceGroup\": \"vm01\", \r\n \"Name\": \"vm01\"\r\n },\r\n {\r\n \"ResourceGroup\": \"vm02\", \r\n \"Name\": \"vm02\"\r\n }\r\n ]"}
```

The screenshot shows the Azure Automation Runbook Test interface. On the left, there's a sidebar with 'Parameters' containing a red box around the 'WEBHOOKDATA' field which contains the JSON provided above. Below it are 'Run Settings' (set to 'Azure') and 'Activity-level tracing' (disabled). On the right, the run status is 'Completed' in green. The log output shows the runbook authenticating with a service principal and certificate, connecting to Azure, and starting two VMs ('vm01' and 'vm02').

NOTE

Azure Automation logs the values of all input parameters with the runbook job. Thus any input provided by the client in the webhook request is logged and available to anyone with access to the automation job. For this reason, you should be cautious about including sensitive information in webhook calls.

Webhook security

The security of a webhook relies on the privacy of its URL, which contains a security token that allows the webhook to be invoked. Azure Automation doesn't perform any authentication on a request as long as it's made to the correct URL. For this reason, your clients shouldn't use webhooks for runbooks that perform highly sensitive operations without using an alternate means of validating the request.

Consider the following strategies:

- You can include logic within a runbook to determine if it's called by a webhook. Have the runbook check the `WebhookName` property of the `WebhookData` parameter. The runbook can perform further validation by looking for particular information in the `RequestHeader` and `RequestBody` properties.
- Have the runbook perform some validation of an external condition when it receives a webhook request. For example, consider a runbook that is called by GitHub any time there's a new commit to a GitHub repository. The runbook might connect to GitHub to validate that a new commit has occurred before continuing.
- Azure Automation supports Azure virtual network service tags, specifically [GuestAndHybridManagement](#). You can use service tags to define network access controls on [network security groups](#) or [Azure Firewall](#) and trigger webhooks from within your virtual network. Service tags can be used in place of specific IP addresses when you create security rules. By specifying the service tag name [GuestAndHybridManagement](#) in the appropriate source or destination field of a rule, you can allow or deny the traffic for the Automation service. This service tag doesn't support allowing more granular control by restricting IP ranges to a specific region.

Create a webhook

A webhook requires a published runbook. This walk through uses a modified version of the runbook created from [Create an Azure Automation runbook](#). To follow along, edit your PowerShell runbook with the following code:

```
param
(
    [Parameter(Mandatory=$false)]
    [object] $WebhookData
)

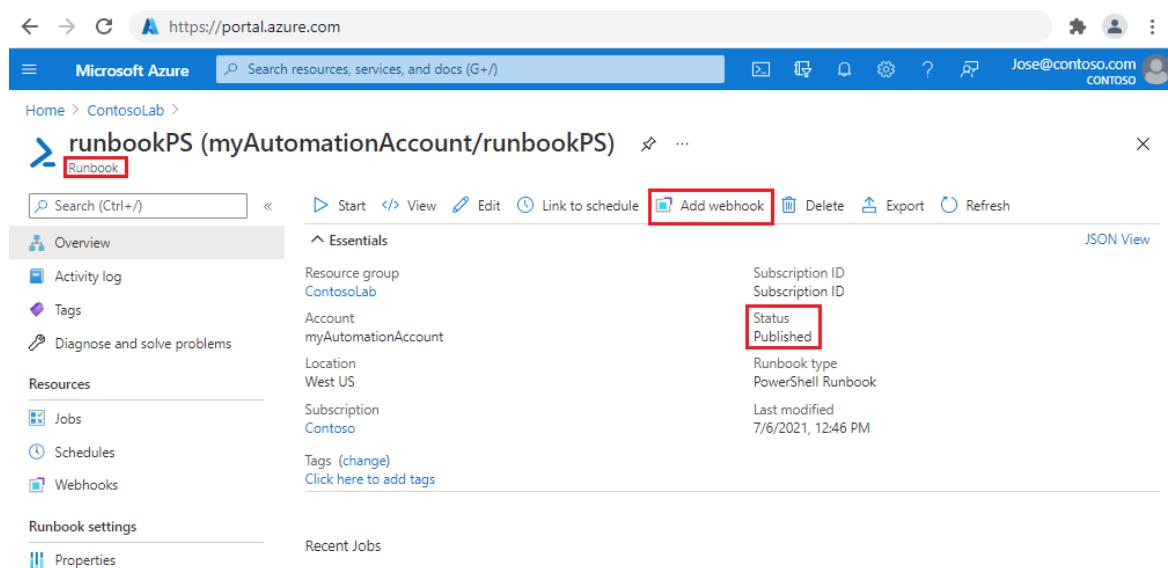
if ($WebhookData.RequestBody) {
    $names = (ConvertFrom-Json -InputObject $WebhookData.RequestBody)

    foreach ($x in $names)
    {
        $name = $x.Name
        Write-Output "Hello $name"
    }
}
else {
    Write-Output "Hello World!"
}
```

Then save and publish the revised runbook. The examples below show to create a webhook using the Azure portal, PowerShell, and REST.

From the portal

1. Sign in to the [Azure portal](#).
2. In the Azure portal, navigate to your Automation account.
3. Under **Process Automation**, select **Runbooks** to open the **Runbooks** page.
4. Select your runbook from the list to open the **Runbook Overview** page.
5. Select **Add webhook** to open the **Add Webhook** page.



The screenshot shows the Azure Runbook Overview page for a runbook named "runbookPS". The "Add webhook" button is highlighted with a red box. Other highlighted fields include "Status Published" and "Subscription ID".

Essentials	
Resource group ContosoLab	Subscription ID
Account myAutomationAccount	Subscription ID
Location West US	Status Published
Subscription Contoso	Runbook type PowerShell Runbook
Tags Click here to add tags	Last modified 7/6/2021, 12:46 PM

6. On the **Add Webhook** page, select **Create new webhook**.

Add Webhook

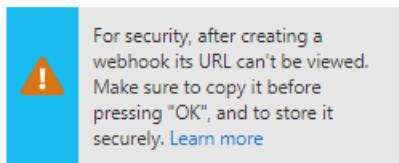
Start a runbook via a simple HTTP POST to a URL

Webhook >
Create new webhook

Parameters and run settings >
Configure parameters and run settings

7. Enter in the **Name** for the webhook. The expiration date for the field **Expires** defaults to one year from the current date.
8. Click the copy icon or press **Ctrl+C** to copy the URL of the webhook. Then save the URL to a secure location.

Create a new webhook



Name *

Enter the webhook name...

Enabled *

Yes No

Expires *

07/06/2022 1:03:01 PM

URL

<https://ad7f1818-7ea9-4567-b43a-b15b...>

OK

IMPORTANT

Once you create the webhook, you cannot retrieve the URL again. Make sure you copy and record it as above.

9. Select **OK** to return to the **Add Webhook** page.
10. From the **Add Webhook** page, select **Configure parameters and run settings** to open the **Parameters** page.

Add Webhook

The screenshot shows the 'Add Webhook' wizard. Step 1: Start a runbook via a simple HTTP POST to a URL. Step 2: Webhook > Create new webhook. Step 3: Parameters and run settings > Configure parameters and run settings. The third step is highlighted with a red box.

11. Review the **Parameters** page. For the example runbook used in this article, no changes are needed. Select **OK** to return to the **Add Webhook** page.
12. From the **Add Webhook** page, select **Create**. The webhook is created and you're returned to the **Runbook Overview** page.

Using PowerShell

1. Verify you have the latest version of the PowerShell [Az Module](#) installed.
2. Sign in to Azure interactively using the [Connect-AzAccount](#) cmdlet and follow the instructions.

```
# Sign in to your Azure subscription
$sub = Get-AzSubscription -ErrorAction SilentlyContinue
if(-not($sub))
{
    Connect-AzAccount
}
```

3. Use the [New-AzAutomationWebhook](#) cmdlet to create a webhook for an Automation runbook. Provide an appropriate value for the variables and then execute the script.

```
# Initialize variables with your relevant values
$resourceGroup = "resourceGroupName"
$automationAccount = "automationAccountName"
$runbook = "runbookName"
$psWebhook = "webhookName"

# Create webhook
$newWebhook = New-AzAutomationWebhook ` 
    -ResourceGroup $resourceGroup ` 
    -AutomationAccountName $automationAccount ` 
    -Name $psWebhook ` 
    -RunbookName $runbook ` 
    -IsEnabled $True ` 
    -ExpiryTime "12/31/2022" ` 
    -Force

# Store URL in variable; reveal variable
$uri = $newWebhook.WebhookURI
$uri
```

The output will be a URL that looks similar to:

```
https://ad7f1818-7ea9-4567-b43a.webhook.wus.azure-automation.net/webhooks?
token=uTi69VZ4RCa42zfKHCeHmJa2W9fd
```

4. You can also verify the webhook with the PowerShell cmdlet [Get-AzAutomationWebhook](#).

```
Get-AzAutomationWebhook ` 
    -ResourceGroup $resourceGroup ` 
    -AutomationAccountName $automationAccount ` 
    -Name $psWebhook
```

Using REST

The PUT command is documented at [Webhook - Create Or Update](#). This example uses the PowerShell cmdlet [Invoke-RestMethod](#) to send the PUT request.

1. Create a file called `webhook.json` and then paste the following code:

```
{
  "name": "RestWebhook",
  "properties": {
    "isEnabled": true,
    "expiryTime": "2022-03-29T22:18:13.7002872Z",
    "runbook": {
      "name": "runbookName"
    }
  }
}
```

Before running, modify the value for the `runbook:name` property with the actual name of your runbook. Review [Webhook properties](#) for more information about these properties.

2. Verify you have the latest version of the PowerShell [Az Module](#) installed.
3. Sign in to Azure interactively using the [Connect-AzAccount](#) cmdlet and follow the instructions.

```
# Sign in to your Azure subscription
$sub = Get-AzSubscription -ErrorAction SilentlyContinue
if(-not($sub))
{
    Connect-AzAccount
}
```

4. Provide an appropriate value for the variables and then execute the script.

```
# Initialize variables
$subscription = "subscriptionID"
$resourceGroup = "resourceGroup"
$automationAccount = "automationAccount"
$runbook = "runbookName"
$restWebhook = "webhookName"
$file = "path\webhook.json"

# consume file
$body = Get-Content $file

# Craft Uri
$restURI =
"https://management.azure.com/subscriptions/$subscription/resourceGroups/$resourceGroup/providers/Microsoft.Automation/automationAccounts/$automationAccount/webhooks/$restWebhook`?api-version=2015-10-31"
```

5. Run the following script to obtain an access token. If your access token expired, you need to rerun the script.

```

# Obtain access token
$azContext = Get-AzContext
$azProfile =
[Microsoft.Azure.Commands.Common.Authentication.Abstractions.AzureRmProfileProvider]::Instance.Profile
$profileClient = New-Object -TypeName Microsoft.Azure.Commands.ResourceManager.Common.RMProfileClient
-ArgumentList ($azProfile)
$token = $profileClient.AcquireAccessToken($azContext.Subscription.TenantId)
$authHeader = @{
    'Content-Type'='application/json'
    'Authorization'='Bearer ' + $token.AccessToken
}

```

6. Run the following script to create the webhook using the REST API.

```

# Invoke the REST API
# Store URL in variable; reveal variable
$response = Invoke-RestMethod -Uri $restURI -Method Put -Headers $authHeader -Body $body
$webhookURI = $response.properties.uri
$webhookURI

```

The output is a URL that looks similar to:

```
https://ad7f1818-7ea9-4567-b43a.azure-automation.net/webhooks?
token=uT169VZ4RCa42zfKHCeHmJa2W9fd
```

7. You can also use [Webhook - Get](#) to retrieve the webhook identified by its name. You can run the following PowerShell commands:

```
$response = Invoke-RestMethod -Uri $restURI -Method GET -Headers $authHeader
$response | ConvertTo-Json
```

Use a webhook

This example uses the PowerShell cmdlet [Invoke-WebRequest](#) to send the POST request to your new webhook.

1. Prepare values to pass to the runbook as the body for the webhook call. For relatively simple values, you could script the values as follows:

```

$Names = @(
    @{
        Name="Hawaii"
    },
    @{
        Name="Seattle"
    },
    @{
        Name="Florida"
    }
)

$body = ConvertTo-Json -InputObject $Names

```

2. For larger sets, you may wish to use a file. Create a file named `names.json` and then paste the following code::

```
[
    {
        "Name": "Hawaii"
    },
    {
        "Name": "Florida"
    },
    {
        "Name": "Seattle"
    }
]
```

Change the value for the variable `$file` with the actual path to the json file before running the following PowerShell commands.

```
# Revise file path with actual path
$file = "path\names.json"
$bodyFile = Get-Content -Path $file
```

- Run the following PowerShell commands to call the webhook using the REST API.

```
$response = Invoke-WebRequest -Method Post -Uri $webhookURI -Body $body -UseBasicParsing
$response

$responseFile = Invoke-WebRequest -Method Post -Uri $webhookURI -Body $bodyFile -UseBasicParsing
$responseFile
```

For illustrative purposes, two calls were made for the two different methods of producing the body. For production, use only one method. The output should look similar as follows (only one output is shown):

```
StatusCode      : 202
StatusDescription : Accepted
Content         : {[JobId:"["0707920e8-96c7-44a3-9976-afb9ce0db982"]"]}
RawContent      : HTTP/1.1 202 Accepted
Pragma          : no-cache
Strict-Transport-Security: max-age=31536000; includeSubDomains
Content-Length: 51
Cache-Control: no-cache
Content-Type: application/json; charset=utf-8
Date...
Forms           :
Headers         : {[Pragma, no-cache], [Strict-Transport-Security, max-age=31536000; includeSubDomains], [Content-Length, 51], [Cache-Control, no-cache]...}
Images          :
InputFields     :
Links           :
ParsedHtml      :
RawContentLength: 51
```

The client receives one of the following return codes from the `POST` request.

CODE	TEXT	DESCRIPTION
202	Accepted	The request was accepted, and the runbook was successfully queued.
400	Bad Request	The request wasn't accepted for one of the following reasons: <ul style="list-style-type: none"> The webhook has expired. The webhook is disabled. The token in the URL is invalid.
404	Not Found	The request wasn't accepted for one of the following reasons: <ul style="list-style-type: none"> The webhook wasn't found. The runbook wasn't found. The account wasn't found.
500	Internal Server Error	The URL was valid, but an error occurred. Resubmit the request.

Assuming the request is successful, the webhook response contains the job ID in JSON format as shown below. It contains a single job ID, but the JSON format allows for potential future enhancements.

```
{"JobIds":["<JobId>"]}
```

- The PowerShell cmdlet `Get-AzAutomationJobOutput` will be used to get the output. The [Azure Automation API](#) could also be used.

```
#isolate job ID
$jobid = (ConvertFrom-Json ($response.Content)).jobids[0]

# Get output
Get-AzAutomationJobOutput `-
    -AutomationAccountName $automationAccount `-
    -Id $jobid `-
    -ResourceGroupName $resourceGroup `-
    -Stream Output
```

The output should look similar to the following:

```
ResourceGroupName : ContosoLab
AutomationAccountName : myAutomationAccount
JobId : 22c3c9c8-33e5-4f00-81a4-f2f16d5be3d7
StreamRecordId : 22c3c9c8-33e5-4f00-81a4-f2f16d5be3d7_00637612083228105379_0000000000000000000005
Time : 7/6/2021 10:45:22 PM +00:00
Summary : Hello Hawaii
Type : Output

ResourceGroupName : ContosoLab
AutomationAccountName : myAutomationAccount
JobId : 22c3c9c8-33e5-4f00-81a4-f2f16d5be3d7
StreamRecordId : 22c3c9c8-33e5-4f00-81a4-f2f16d5be3d7_00637612083228905818_00000000000000000006
Time : 7/6/2021 10:45:22 PM +00:00
Summary : Hello Seattle
Type : Output

ResourceGroupName : ContosoLab
AutomationAccountName : myAutomationAccount
JobId : 22c3c9c8-33e5-4f00-81a4-f2f16d5be3d7
StreamRecordId : 22c3c9c8-33e5-4f00-81a4-f2f16d5be3d7_00637612083229605419_00000000000000000007
Time : 7/6/2021 10:45:22 PM +00:00
Summary : Hello Florida
Type : Output
```

Update a webhook

When a webhook is created, it has a validity time period of 10 years, after which it automatically expires. Once a webhook has expired, you can't reactivate it. You can only remove and then recreate it. You can extend a webhook that hasn't reached its expiration time. To extend a webhook, perform the following steps.

1. Navigate to the runbook that contains the webhook.
2. Under **Resources**, select **Webhooks**, and then the webhook that you want to extend.
3. From the **Webhook** page, choose a new expiration date and time and then select **Save**.

Review the API call [Webhook - Update](#) and PowerShell cmdlet [Set-AzAutomationWebhook](#) for other possible modifications.

Clean up resources

Here are examples of removing a webhook from an Automation runbook.

- Using PowerShell, the [Remove-AzAutomationWebhook](#) cmdlet can be used as shown below. No output is returned.

```
Remove-AzAutomationWebhook `-
    -ResourceGroup $resourceGroup `-
    -AutomationAccountName $automationAccount `-
    -Name $psWebhook
```

- Using REST, the REST [Webhook - Delete](#) API can be used as shown below.

```
Invoke-WebRequest -Method Delete -Uri $restURI -Headers $authHeader
```

An output of `StatusCode : 200` means a successful deletion.

Create runbook and webhook with ARM template

Automation webhooks can also be created using [Azure Resource Manager](#) templates. This sample template creates an Automation account, four runbooks, and a webhook for the named runbook.

1. Create a file named `webhook_deploy.json` and then paste the following code:

```
{  
    "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",  
    "contentVersion": "1.0.0.0",  
    "parameters": {  
        "automationAccountName": {  
            "type": "String",  
            "metadata": {  
                "description": "Automation account name"  
            }  
        },  
        "webhookName": {  
            "type": "String",  
            "metadata": {  
                "description": "Webhook Name"  
            }  
        },  
        "runbookName": {  
            "type": "String",  
            "metadata": {  
                "description": "Runbook Name for which webhook will be created"  
            }  
        },  
        "WebhookExpiryTime": {  
            "type": "String",  
            "metadata": {  
                "description": "Webhook Expiry time"  
            }  
        },  
        "_artifactsLocation": {  
            "defaultValue": "https://raw.githubusercontent.com/Azure/azure-quickstart-  
templates/master/quickstarts/microsoft.automation/101-automation/",  
            "type": "String",  
            "metadata": {  
                "description": "URI to artifacts location"  
            }  
        }  
    },  
    "resources": [  
        {  
            "type": "Microsoft.Automation/automationAccounts",  
            "apiVersion": "2020-01-13-preview",  
            "name": "[parameters('automationAccountName')]",  
            "location": "[resourceGroup().location]",  
            "properties": {  
                "sku": {  
                    "name": "Free"  
                }  
            },  
            "resources": [  
                {  
                    "type": "runbooks",  
                    "apiVersion": "2018-06-30",  
                    "name": "[parameters('runbookName')]",  
                    "location": "[resourceGroup().location]",  
                    "dependsOn": [  
                        "[parameters('automationAccountName')]"  
                    ],  
                    "properties": {  
                        "scriptContent": "Write-Host 'Hello from PowerShell Runbook'"  
                    }  
                }  
            ]  
        }  
    ]  
}
```

```

    "properties": {
        "runbookType": "Python2",
        "logProgress": "false",
        "logVerbose": "false",
        "description": "Sample Runbook",
        "publishContentLink": {
            "uri": "[uri(parameters('_artifactsLocation'),
'scripts/AzureAutomationTutorialPython2.py'))]",
            "version": "1.0.0.0"
        }
    },
    {
        "type": "webhooks",
        "apiVersion": "2018-06-30",
        "name": "[parameters('webhookName')]",
        "dependsOn": [
            "[parameters('automationAccountName')]",
            "[parameters('runbookName')]"
        ],
        "properties": {
            "isEnabled": true,
            "expiryTime": "[parameters('WebhookExpiryTime')]",
            "runbook": {
                "name": "[parameters('runbookName')]"
            }
        }
    }
],
"outputs": {
    "webhookUri": {
        "type": "String",
        "value": "[reference(parameters('webhookName')).uri]"
    }
}
}

```

- The following PowerShell code sample deploys the template from your machine. Provide an appropriate value for the variables and then execute the script.

```

$resourceGroup = "resourceGroup"
$templateFile = "path\ webhook_deploy.json"
$armAutomationAccount = "automationAccount"
$armRunbook = "ARMrunbookName"
$armWebhook = "webhookName"
$webhookExpiryTime = "12-31-2022"

New-AzResourceGroupDeployment ` 
    -Name "testDeployment" ` 
    -ResourceGroupName $resourceGroup ` 
    -TemplateFile $templateFile ` 
    -automationAccountName $armAutomationAccount ` 
    -runbookName $armRunbook ` 
    -webhookName $armWebhook ` 
    -WebhookExpiryTime $webhookExpiryTime

```

NOTE

For security reasons, the URI is only returned the first time a template is deployed.

Next steps

- To trigger a runbook from an alert, see [Use an alert to trigger an Azure Automation runbook](#).

Use the Azure Automation graphical runbook SDK (preview)

11/2/2020 • 3 minutes to read • [Edit Online](#)

Graphical runbooks help manage the complexities of the underlying Windows PowerShell or PowerShell Workflow code. The Microsoft Azure Automation graphical authoring SDK enables developers to create and edit graphical runbooks for use with Azure Automation. This article describes basic steps in creating a graphical runbook from your code.

Prerequisites

Import the `Orchestrator.GraphRunbook.Model.dll` package by downloading the [SDK](#).

Create a runbook object instance

Reference the `Orchestrator.GraphRunbook.Model` assembly and create an instance of the `Orchestrator.GraphRunbook.Model.GraphRunbook` class:

```
using Orchestrator.GraphRunbook.Model;
using Orchestrator.GraphRunbook.Model.ExecutableView;

var runbook = new GraphRunbook();
```

Add runbook parameters

Instantiate `Orchestrator.GraphRunbook.Model.Parameter` objects and add them to the runbook:

```
runbook.AddParameter(
    new Parameter("YourName") {
        TypeName = typeof(string).FullName,
        DefaultValue = "World",
        Optional = true
    });

runbook.AddParameter(
    new Parameter("AnotherParameter") {
        TypeName = typeof(int).FullName, ...
    });
```

Add activities and links

Instantiate activities of appropriate types and add them to the runbook:

```

var writeOutputActivityType = new CommandActivityType {
    CommandName = "Write-Output",
    ModuleName = "Microsoft.PowerShell.Utility",
    InputParameterSets = new [] {
        new ParameterSet {
            Name = "Default",
            new [] {
                new Parameter("InputObject"), ...
            }
        },
        ...
    }
};

var outputName = runbook.AddActivity(
    new CommandActivity("Output name", writeOutputActivityType) {
        ParameterSetName = "Default",
        Parameters = new ActivityParameters {
            {
                "InputObject",
                new RunbookParameterValueDescriptor("YourName")
            }
        },
        PositionX = 0,
        PositionY = 0
    });
};

var initializeRunbookVariable = runbook.AddActivity(
    new WorkflowScriptActivity("Initialize variable") {
        Process = "$a = 0",
        PositionX = 0,
        PositionY = 100
    });

```

Activities are implemented by the following classes in the `Orchestrator.GraphRunbook.Model` namespace.

CLASS	ACTIVITY
CommandActivity	Invokes a PowerShell command (cmdlet, function, etc.).
InvokeRunbookActivity	Invokes another runbook inline.
JunctionActivity	Waits for all incoming branches to finish.
WorkflowScriptActivity	Executes a block of PowerShell or PowerShell Workflow code (depending on the runbook type) in the context of the runbook. This is a powerful tool, but do not overuse it: the UI will show this script block as text; the execution engine will treat the provided block as a black box, and will make no attempts to analyze its content, except for a basic syntax check. If you just need to invoke a single PowerShell command, prefer CommandActivity.

NOTE

Don't derive your own activities from the provided classes. Azure Automation can't use runbooks with custom activity types.

You must provide `CommandActivity` and `InvokeRunbookActivity` parameters as value descriptors, not direct values. Value descriptors specify how to produce the actual parameter values. The following value descriptors

are currently provided:

Descriptor	Definition
ConstantValueDescriptor	Refers to a hard-coded constant value.
RunbookParameterValueDescriptor	Refers to a runbook parameter by name.
ActivityOutputValueDescriptor	Refers to an upstream activity output, allowing one activity to "subscribe" to data produced by another activity.
AutomationVariableValueDescriptor	Refers to an Automation Variable asset by name.
AutomationCredentialValueDescriptor	Refers to an Automation Certificate asset by name.
AutomationConnectionValueDescriptor	Refers to an Automation Connection asset by name.
PowerShellExpressionValueDescriptor	Specifies a free-form PowerShell expression that will be evaluated just before invoking the activity. This is a powerful tool, but do not overuse it: the UI will show this expression as text; the execution engine will treat the provided block as a black box, and will make no attempts to analyze its content, except for a basic syntax check. When possible, prefer more specific value descriptors.

NOTE

Don't derive your own value descriptors from the provided classes. Azure Automation can't use runbooks with custom value descriptor types.

Instantiate links connecting activities and add them to the runbook:

```
runbook.AddLink(new Link(activityA, activityB, LinkType.Sequence));  
  
runbook.AddLink(  
    new Link(activityB, activityC, LinkType.Pipeline) {  
        Condition = string.Format("${ActivityOutput['{0}']} -gt 0", activityB.Name)  
    });
```

Save the runbook to a file

Use `Orchestrator.GraphRunbook.Model.Serialization.RunbookSerializer` to serialize a runbook to a string:

```
var serialized = RunbookSerializer.Serialize(runbook);
```

You can save this string to a file with the `.graphrunbook` extension. The corresponding runbook can be imported into Azure Automation. The serialized format might change in the future versions of `Orchestrator.GraphRunbook.Model.dll`. We promise backward compatibility: any runbook serialized with an older version of `Orchestrator.GraphRunbook.Model.dll` can be deserialized by any newer version. Forward compatibility is not guaranteed: a runbook serialized with a newer version may not be deserializable by older versions.

Next steps

For more information, see [Author graphical runbooks in Azure Automation](#).

Configure runbook output and message streams

4/28/2021 • 13 minutes to read • [Edit Online](#)

Most Azure Automation runbooks have some form of output. This output can be an error message to the user or a complex object intended to be used with another runbook. Windows PowerShell provides [multiple streams](#) to send output from a script or workflow. Azure Automation works with each of these streams differently. You should follow best practices for using the streams when you're creating a runbook.

The following table briefly describes each stream with its behavior in the Azure portal for published runbooks and during [testing of a runbook](#). The output stream is the main stream used for communication between runbooks. The other streams are classified as message streams, intended to communicate information to the user.

STREAM	DESCRIPTION	PUBLISHED	TEST
Error	Error message intended for the user. Unlike with an exception, the runbook continues after an error message by default.	Written to job history	Displayed in Test output pane
Debug	Messages intended for an interactive user. Shouldn't be used in runbooks.	Not written to job history	Not displayed in Test output pane
Output	Objects intended to be consumed by other runbooks.	Written to job history	Displayed in Test output pane
Progress	Records automatically generated before and after each activity in the runbook. The runbook shouldn't try to create its own progress records, since they're intended for an interactive user.	Written to job history only if progress logging is turned on for the runbook	Not displayed in Test output pane
Verbose	Messages that give general or debugging information.	Written to job history only if verbose logging is turned on for the runbook	Displayed in Test output pane only if <code>VerbosePreference</code> variable is set to Continue in runbook
Warning	Warning message intended for the user.	Written to job history	Displayed in Test output pane

Use the output stream

The output stream is used for the output of objects created by a script or workflow when it runs correctly. Azure Automation primarily uses this stream for objects to be consumed by parent runbooks that call the [current runbook](#). When a parent [calls a runbook inline](#), the child returns data from the output stream to the parent.

Your runbook uses the output stream to communicate general information to the client only if it is never called by another runbook. As a best practice, however, you runbooks should typically use the [verbose stream](#) to communicate general information to the user.

Have your runbook write data to the output stream using [Write-Output](#). Alternatively, you can put the object on its own line in the script.

```
#The following lines both write an object to the output stream.  
Write-Output -InputObject $object  
$object
```

Handle output from a function

When a runbook function writes to the output stream, the output is passed back to the runbook. If the runbook assigns that output to a variable, the output is not written to the output stream. Writing to any other streams from within the function writes to the corresponding stream for the runbook. Consider the following sample PowerShell Workflow runbook.

```
Workflow Test-Runbook  
{  
    Write-Verbose "Verbose outside of function" -Verbose  
    Write-Output "Output outside of function"  
    $functionOutput = Test-Function  
    $functionOutput  
  
    Function Test-Function  
{  
        Write-Verbose "Verbose inside of function" -Verbose  
        Write-Output "Output inside of function"  
    }  
}
```

The output stream for the runbook job is:

```
Output inside of function  
Output outside of function
```

The verbose stream for the runbook job is:

```
Verbose outside of function  
Verbose inside of function
```

Once you've published the runbook and before you start it, you must also turn on verbose logging in the runbook settings to get the verbose stream output.

Declare output data type

The following are examples of output data types:

- `System.String`
- `System.Int32`
- `System.Collections.Hashtable`
- `Microsoft.Azure.Commands.Compute.Models.PSVirtualMachine`

Declare output data type in a workflow

A workflow specifies the data type of its output using the [OutputType attribute](#). This attribute has no effect during runtime, but it provides you an indication at design time of the expected output of the runbook. As the

tool set for runbooks continues to evolve, the importance of declaring output data types at design time increases. Therefore it's a best practice to include this declaration in any runbooks that you create.

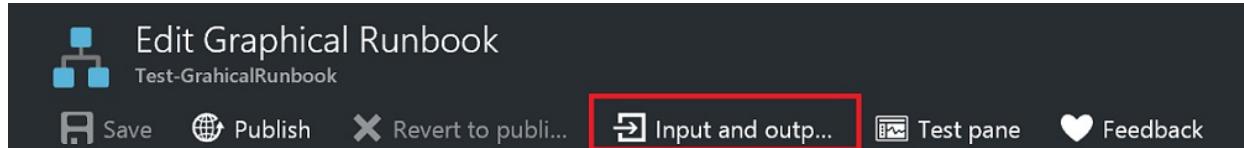
The following sample runbook outputs a string object and includes a declaration of its output type. If your runbook outputs an array of a certain type, then you should still specify the type as opposed to an array of the type.

```
Workflow Test-Runbook
{
    [OutputType([string])]

    $output = "This is some string output."
    Write-Output $output
}
```

Declare output data type in a graphical runbook

To declare an output type in a graphical or graphical PowerShell Workflow runbook, you can select the **Input** and **Output** menu option and enter the output type. It's recommended to use the full .NET class name to make the type easily identifiable when a parent runbook references it. Using the full name exposes all the properties of the class to the databus in the runbook and increases flexibility when the properties are used for conditional logic, logging, and referencing as values for other runbook activities.

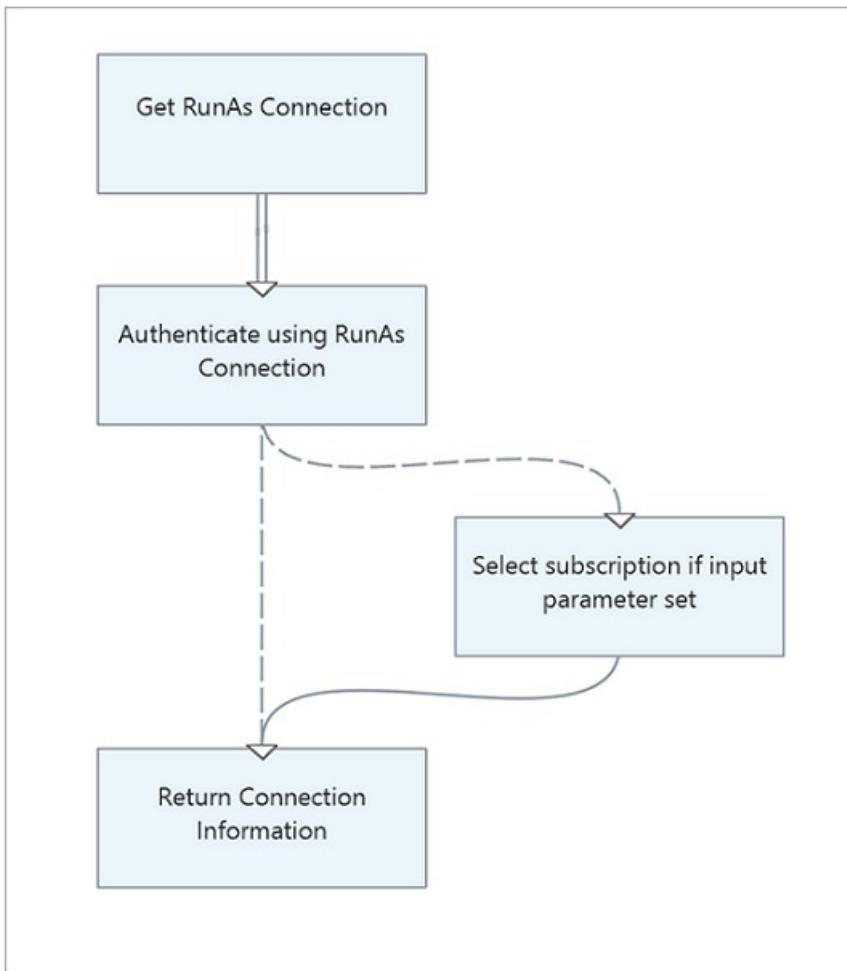


NOTE

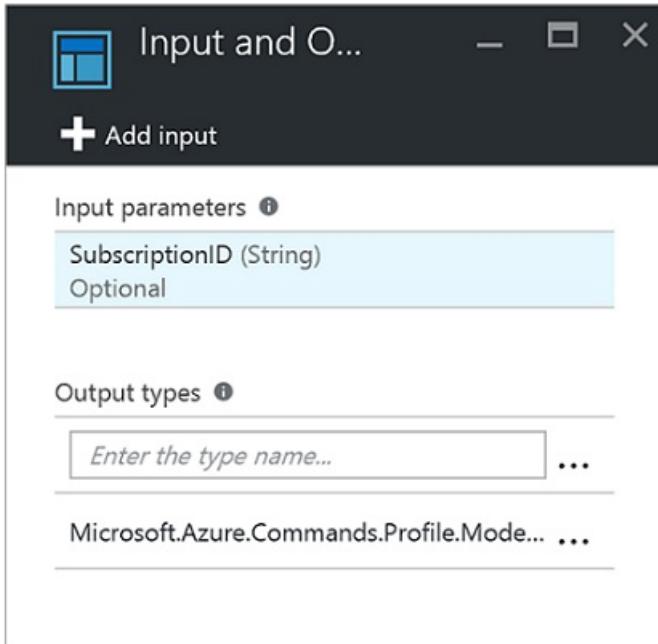
After you enter a value in the **Output Type** field in the Input and Output properties pane, be sure to click outside the control so that it recognizes your entry.

The following example shows two graphical runbooks to demonstrate the Input and Output feature. Applying the modular runbook design model, you have one runbook as the Authenticate Runbook template managing authentication with Azure using the Run As account. The second runbook, which normally performs core logic to automate a given scenario, in this case executes the Authenticate Runbook template. It displays the results to your Test output pane. Under normal circumstances, you would have this runbook do something against a resource leveraging the output from the child runbook.

Here is the basic logic of the **AuthenticateTo-Azure** runbook.

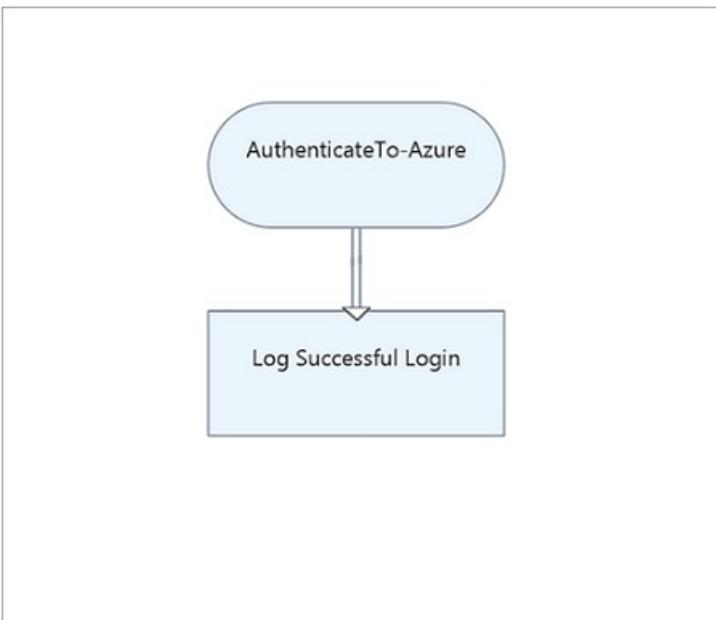


The runbook includes the output type `Microsoft.Azure.Commands.Profile.Models.PSAzureContext`, which returns the authentication profile properties.



While this runbook is straightforward, there is one configuration item to call out here. The last activity executes the `Write-Output` cmdlet to write profile data to a variable using a PowerShell expression for the `InputObject` parameter. This parameter is required for `Write-Output`.

The second runbook in this example, named **Test-ChildOutputType**, simply defines two activities.



The first activity calls the **AuthenticateTo-Azure** runbook. The second activity runs the `Write-Verbose` cmdlet with Data source set to **Activity output**. Also, **Field path** is set to `Context.Subscription.SubscriptionName`, the context output from the **AuthenticateTo-Azure** runbook.

Activity Parameter Con...		Parameter Value
Name Write-Verbose	<input checked="" type="checkbox"/> MESSAGE AuthenticateTo-Azure (Activity output)	Data source Activity output
* Label <input type="text" value="Log Successful Login"/>		Select data
Comment <input type="text"/>		Selected activity name AuthenticateTo-Azure
Parameters 1 of 1 configured		Field path Context.Subscription.SubscriptionName
Optional additional parameters Configure parameters		
Retry behavior Configure retry behavior		

The resulting output is the name of the subscription.

Test	
Test-ChildOutputType	
Start Stop Suspend Resume	Completed
Parameters No input parameters	Microsoft Azure Internal Consumption
Run Settings Run on Azure	

Working with message streams

Unlike the output stream, message streams communicate information to the user. There are multiple message

streams for different kinds of information, and Azure Automation handles each stream differently.

Write output to warning and error streams

The warning and error streams log problems that occur in a runbook. Azure Automation writes these streams to the job history when executing a runbook. Automation includes the streams in the Test output pane in the Azure portal when a runbook is tested.

By default, a runbook continues to execute after a warning or error. You can specify that your runbook should suspend on a warning or error by having the runbook set a [preference variable](#) before creating the message. For example, to cause the runbook to suspend on an error as it does on an exception, set the `ErrorActionPreference` variable to Stop.

Create a warning or error message using the [Write-Warning](#) or [Write-Error](#) cmdlet. Activities can also write to the warning and error streams.

```
#The following lines create a warning message and then an error message that will suspend the runbook.

$ErrorActionPreference = "Stop"
Write-Warning -Message "This is a warning message."
Write-Error -Message "This is an error message that will stop the runbook because of the preference
variable."
```

Write output to debug stream

Azure Automation uses the debug message stream for interactive users. By default Azure Automation does not capture any debug stream data, only output, error, and warning data are captured as well as verbose data if the runbook is configured to capture it.

In order to capture debug stream data, you have to perform two actions in your runbooks:

1. Set the variable `$GLOBAL:DebugPreference="Continue"`, which tells PowerShell to continue whenever a debug message is encountered. The `$GLOBAL:` portion tells PowerShell to do this in the global scope rather than whatever local scope the script is in at the time the statement is executed.
2. Redirect the debug stream that we don't capture to a stream that we do capture such as `output`. This is done by setting PowerShell redirection against the statement to be executed. For more information on PowerShell redirection, see [About Redirection](#).

Examples

In this example, the runbook is configured using the [Write-Output](#) and [Write-Debug](#) cmdlets with the intention of outputting two different streams.

```
Write-Output "This is an output message."
Write-Debug "This is a debug message."
```

If this runbook were to be executed as is, the output pane for the runbook job would stream the following output:

```
This is an output message.
```

In this example, the runbook is configured similar to the previous example, except the statement

`$GLOBAL:DebugPreference="Continue"` is included with the addition of `5>&1` at the end of the [Write-Debug](#) statement.

```
Write-Output "This is an output message."  
$GLOBAL:DebugPreference="Continue"  
Write-Debug "This is a debug message." 5>&1
```

If this runbook were to be executed, the output pane for the runbook job would stream the following output:

```
This is an output message.  
This is a debug message.
```

This occurs because the `$GLOBAL:DebugPreference="Continue"` statement tells PowerShell to display debug messages, and the addition of `5>&1` to the end of the `Write-Debug` statement tells PowerShell to redirect stream 5 (debug) to stream 1 (output).

Write output to verbose stream

The Verbose message stream supports general information about runbook operation. Since the debug stream is not available for a runbook, your runbook should use verbose messages for debug information.

By default, the job history does not store verbose messages from published runbooks, for performance reasons. To store verbose messages, use the Azure portal **Configure** tab with the **Log Verbose Records** setting to configure your published runbooks to log verbose messages. Turn on this option only to troubleshoot or debug a runbook. In most cases, you should keep the default setting of not logging verbose records.

When [testing a runbook](#), verbose messages aren't displayed even if the runbook is configured to log verbose records. To display verbose messages while [testing a runbook](#), you must set the `VerbosePreference` variable to Continue. With that variable set, verbose messages are displayed in the Test output pane of the Azure portal.

The following code creates a verbose message using the [Write-Verbose](#) cmdlet.

```
#The following line creates a verbose message.  
  
Write-Verbose -Message "This is a verbose message."
```

Handle progress records

You can use the **Configure** tab of the Azure portal to configure a runbook to log progress records. The default setting is to not log the records, to maximize performance. In most cases, you should keep the default setting. Turn on this option only to troubleshoot or debug a runbook.

If you enable progress record logging, your runbook writes a record to job history before and after each activity runs. Testing a runbook does not display progress messages even if the runbook is configured to log progress records.

NOTE

The [Write-Progress](#) cmdlet is not valid in a runbook, since this cmdlet is intended for use with an interactive user.

Work with preference variables

You can set certain Windows PowerShell [preference variables](#) in your runbooks to control the response to data sent to different output streams. The following table lists the preference variables that can be used in runbooks, with their default and valid values. Additional values are available for the preference variables when used in Windows PowerShell outside of Azure Automation.

VARIABLE	DEFAULT VALUE	VALID VALUES
<code>WarningPreference</code>	Continue	Stop Continue SilentlyContinue
<code>ErrorActionPreference</code>	Continue	Stop Continue SilentlyContinue
<code>VerbosePreference</code>	SilentlyContinue	Stop Continue SilentlyContinue

The next table lists the behavior for the preference variable values that are valid in runbooks.

VALUE	BEHAVIOR
Continue	Logs the message and continues executing the runbook.
SilentlyContinue	Continues executing the runbook without logging the message. This value has the effect of ignoring the message.
Stop	Logs the message and suspends the runbook.

Retrieve runbook output and messages

Retrieve runbook output and messages in Azure portal

You can view the details of a runbook job in the Azure portal using the **Jobs** tab for the runbook. The job summary displays the input parameters and the [output stream](#), in addition to general information about the job and any exceptions that have occurred. The job history includes messages from the output stream and [warning and error streams](#). It also includes messages from the [verbose stream](#) and [progress records](#) if the runbook is configured to log verbose and progress records.

Retrieve runbook output and messages in Windows PowerShell

In Windows PowerShell, you can retrieve output and messages from a runbook using the `Get-AzAutomationJobOutput` cmdlet. This cmdlet requires the ID of the job and has a parameter called `Stream` in which to specify the stream to retrieve. You can specify a value of Any for this parameter to retrieve all streams for the job.

The following example starts a sample runbook and then waits for it to complete. Once the runbook completes execution, the script collects the runbook output stream from the job.

```

$job = Start-AzAutomationRunbook -ResourceGroupName "ResourceGroup01" ` 
    -AutomationAccountName "MyAutomationAccount" -Name "Test-Runbook"

$doLoop = $true
While ($doLoop) {
    $job = Get-AzAutomationJob -ResourceGroupName "ResourceGroup01" ` 
        -AutomationAccountName "MyAutomationAccount" -Id $job.JobId
    $status = $job.Status
    $doLoop = (($status -ne "Completed") -and ($status -ne "Failed") -and ($status -ne "Suspended") -and
    ($status -ne "Stopped"))
}

Get-AzAutomationJobOutput -ResourceGroupName "ResourceGroup01" ` 
    -AutomationAccountName "MyAutomationAccount" -Id $job.JobId -Stream Output

# For more detailed job output, pipe the output of Get-AzAutomationJobOutput to Get-
AzAutomationJobOutputRecord
Get-AzAutomationJobOutput -ResourceGroupName "ResourceGroup01" ` 
    -AutomationAccountName "MyAutomationAccount" -Id $job.JobId -Stream Any | Get-AzAutomationJobOutputRecord

```

Retrieve runbook output and messages in graphical runbooks

For graphical runbooks, extra logging of output and messages is available in the form of activity-level tracing. There are two levels of tracing: Basic and Detailed. Basic tracing displays the start and end time for each activity in the runbook, plus information related to any activity retries. Some examples are the number of attempts and the start time of the activity. Detailed tracing includes basic tracing features plus logging of input and output data for each activity.

Currently activity-level tracing writes records using the verbose stream. Therefore you must enable verbose logging when you enable tracing. For graphical runbooks with tracing enabled, there's no need to log progress records. Basic tracing serves the same purpose and is more informative.

TIME	TYPE	DETAILS
2/7/2016, 6:56 PM	Verbose	GraphTrace{ Activity:"Write-Verbose", Event:"ActivityStart", Time:"2016-02-08T02:56:20.1373078Z" }
2/7/2016, 6:56 PM	Verbose	GraphTrace{ Activity:"Write-Verbose", Event:"ActivityInput", Time:"2016-02-08T02:56:24.1372592Z", Val...
2/7/2016, 6:56 PM	Verbose	Verbose message
2/7/2016, 6:56 PM	Verbose	GraphTrace{ Activity:"Write-Verbose", Event:"ActivityOutput", Time:"2016-02-08T02:56:24.4653789Z", V...
2/7/2016, 6:56 PM	Verbose	GraphTrace{ Activity:"Write-Verbose", Event:"ActivityEnd", Time:"2016-02-08T02:56:24.6528783Z", Dura...
2/7/2016, 6:56 PM	Verbose	GraphTrace{ Activity:"Write-Warning", Event:"ActivityStart", Time:"2016-02-08T02:56:24.7935039Z" }
2/7/2016, 6:56 PM	Verbose	GraphTrace{ Activity:"Write-Warning", Event:"ActivityInput", Time:"2016-02-08T02:56:24.8872483Z", Val...
2/7/2016, 6:56 PM	Warning	Warning message
2/7/2016, 6:56 PM	Verbose	GraphTrace{ Activity:"Write-Warning", Event:"ActivityOutput", Time:"2016-02-08T02:56:25.0903685Z", ...
2/7/2016, 6:56 PM	Verbose	GraphTrace{ Activity:"Write-Warning", Event:"ActivityEnd", Time:"2016-02-08T02:56:25.1841205Z", Dur...
2/7/2016, 6:56 PM	Verbose	GraphTrace{ Activity:"Write-Error", Event:"ActivityStart", Time:"2016-02-08T02:56:25.2778702Z" }
2/7/2016, 6:56 PM	Verbose	GraphTrace{ Activity:"Write-Error", Event:"ActivityInput", Time:"2016-02-08T02:56:25.3716198Z", Values...
2/7/2016, 6:56 PM	Verbose	GraphTrace{ Activity:"Write-Error", Event:"ActivityOutput", Time:"2016-02-08T02:56:25.4966177Z", Valu...
2/7/2016, 6:56 PM	Error	Error message
2/7/2016, 6:56 PM	Verbose	GraphTrace{ Activity:"Write-Error", Event:"ActivityEnd", Time:"2016-02-08T02:56:26.7778651Z", Duratio...

You can see from the image that enabling verbose logging and tracing for graphical runbooks makes much more information available in the production **Job Streams** view. This extra information can be essential for troubleshooting production problems with a runbook.

However, unless you require this information to track the progress of a runbook for troubleshooting, you might want to keep tracing turned off as a general practice. The trace records can be especially numerous. With graphical runbook tracing, you can get two to four records per activity, depending on your configuration of Basic or Detailed tracing.

To enable activity-level tracing:

1. In the Azure portal, open your Automation account.
2. Select **Runbooks** under **Process Automation** to open the list of runbooks.
3. On the Runbooks page, select a graphical runbook from your list of runbooks.
4. Under **Settings**, click **Logging and tracing**.
5. On the Logging and Tracing page, under **Log verbose records**, click **On** to enable verbose logging.
6. Under **Activity-level tracing**, change the trace level to **Basic** or **Detailed**, based on the level of tracing you require.



Retrieve runbook output and messages in Microsoft Azure Monitor logs

Azure Automation can send runbook job status and job streams to your Log Analytics workspace. Azure Monitor supports logs that allow you to:

- Get insight on your Automation jobs.
- Trigger an email or alert based on your runbook job status, for example, Failed or Suspended.
- Write advanced queries across job streams.
- Correlate jobs across Automation accounts.
- Visualize job history.

For more information about configuring integration with Azure Monitor Logs to collect, correlate, and act on job data, see [Forward job status and job streams from Automation to Azure Monitor Logs](#).

Next steps

- To work with runbooks, see [Manage runbooks in Azure Automation](#).
- If you are unfamiliar with PowerShell scripting, see [PowerShell](#) documentation.
- For the Azure Automation PowerShell cmdlet reference, see [Az.Automation](#).

Handle errors in graphical runbooks

4/22/2021 • 3 minutes to read • [Edit Online](#)

A key design principle to consider for your Azure Automation graphical runbook is the identification of issues that the runbook might experience during execution. These issues can include success, expected error states, and unexpected error conditions.

Often, if there is a non-terminating error that occurs with a runbook activity, Windows PowerShell handles the activity by processing any activity that follows, regardless of the error. The error is likely to generate an exception, but the next activity is still allowed to run.

Your graphical runbook should include error handling code to deal with execution issues. To validate the output of an activity or handle an error, you can use a PowerShell code activity, define conditional logic on the output link of the activity, or apply another method.

Azure Automation graphical runbooks have been improved with the capability to include error handling. You can now turn exceptions into non-terminating errors and create error links between activities. The improved process allows your runbook to catch errors and manage realized or unexpected conditions.

PowerShell error types

The types of PowerShell errors that can occur during runbook execution are terminating errors and non-terminating errors.

Terminating error

A terminating error is a serious error during execution that halts a command or script execution completely. Examples include nonexistent cmdlets, syntax errors that prevent a cmdlet from running, and other fatal errors.

Non-terminating error

A non-terminating error is a non-serious error that allows execution to continue despite the error condition. Examples include operational errors, such as file not found errors and permissions issues.

When to use error handling

Use error handling in your runbook when a critical activity throws an error or exception. It's important to prevent the next activity in the runbook from processing and to handle the error appropriately. Handling the error is especially critical when your runbooks are supporting a business or service operations process.

Add error links

For each activity that can produce an error, you can add an error link pointing to any other activity. The destination activity can be of any type, including code activity, invocation of a cmdlet, invocation of another runbook, and so on. The destination activity can also have outgoing links, either regular or error links. The links allow the runbook to implement complex error handling logic without resorting to a code activity.

The recommended practice is to create a dedicated error handling runbook with common functionality, but this practice isn't mandatory. For example, consider a runbook that tries to start a virtual machine and install an application on it. If the VM doesn't start correctly, it:

1. Sends a notification about this problem.
2. Starts another runbook that automatically provisions a new VM instead.

One solution is to have an error link in the runbook pointing to an activity that handles step one. For example, the runbook can connect the `Write-Warning` cmdlet to an activity for step two, such as the `Start-AzAutomationRunbook` cmdlet.

You can also generalize this behavior for use in many runbooks by putting these two activities in a separate error handling runbook. Before your original runbook calls this error handling runbook, it can construct a custom message from its data and then pass it as a parameter to the error handling runbook.

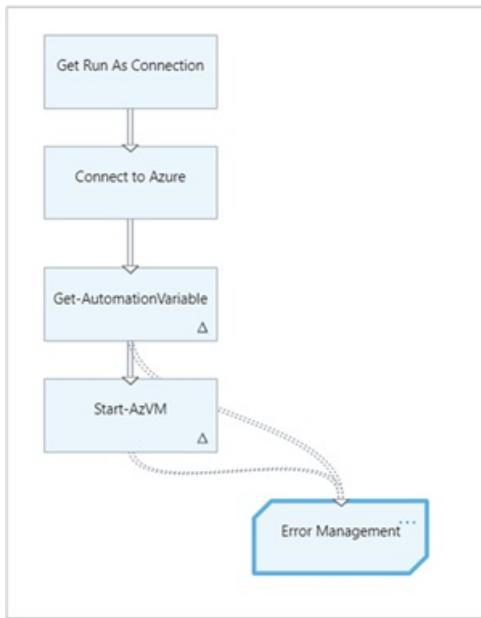
Turn exceptions into non-terminating errors

Each activity in your runbook has a configuration setting that turns exceptions into non-terminating errors. By default, this setting is disabled. We recommend enabling this setting on any activity where your runbook handles errors. This setting ensures that the runbook handles both terminating and non-terminating errors in the activity as non-terminating errors, using an error link.

After enabling the configuration setting, have your runbook create an activity that handles the error. If the activity produces any error, the outgoing error links are followed. The regular links are not followed, even if the activity produces regular output as well.



In the following example, a runbook retrieves a variable that contains the computer name of a VM. It then attempts to start the VM with the next activity.



The `Get-AutomationVariable` activity and the `Start-AzVM` cmdlet are configured to convert exceptions to errors. If there are problems getting the variable or starting the VM, the code generates errors.

Name
Get-AutomationVariable

Label * ⓘ
 ✓

Comment

Convert exceptions to errors ⓘ
 Yes No

Parameters >
[Configure parameter values](#)

Optional additional parameters >
[Configure parameters](#)

Retry behavior >
[Configure retry behavior](#)

Notices
⚠ Exceptions converted to errors

Error links flow from these activities to a single `error management` code activity. This activity is configured with a simple PowerShell expression that uses the `throw` keyword to stop processing, along with `$Error.Exception.Message` to get the message that describes the current exception.

Code Editor

PowerShell code ⓘ

```
Throw "Error occurred with an activity." + "Error returned is: " $Error.Exception.Message
```

OK

The screenshot shows a 'Code Editor' window with a single line of PowerShell code. The code is as follows:

```
Throw "Error occurred with an activity." + "Error returned is: " $Error.Exception.Message
```

The 'OK' button at the bottom left indicates the code has been saved or accepted.

Next steps

- For information about resolving graphical runbook errors, see [Troubleshoot runbook issues](#).

Forward Azure Automation job data to Azure Monitor logs

4/22/2021 • 7 minutes to read • [Edit Online](#)

Azure Automation can send runbook job status and job streams to your Log Analytics workspace. This process does not involve workspace linking and is completely independent. Job logs and job streams are visible in the Azure portal, or with PowerShell, for individual jobs and this allows you to perform simple investigations. Now with Azure Monitor logs you can:

- Get insight into the status of your Automation jobs.
- Trigger an email or alert based on your runbook job status (for example, failed or suspended).
- Write advanced queries across your job streams.
- Correlate jobs across Automation accounts.
- Use custom views and search queries to visualize your runbook results, runbook job status, and other related key indicators or metrics.

Prerequisites

To start sending your Automation logs to Azure Monitor logs, you need:

- The latest release of [Azure PowerShell](#).
- A Log Analytics workspace and its resource ID. For more information, see [Get started with Azure Monitor logs](#).
- The resource ID of your Azure Automation account.

How to find resource IDs

1. Use the following command to find the resource ID for your Azure Automation account:

```
# Find the ResourceId for the Automation account
Get-AzResource -ResourceType "Microsoft.Automation/automationAccounts"
```

2. Copy the value for **ResourceId**.

3. Use the following command to find the resource ID of your Log Analytics workspace:

```
# Find the ResourceId for the Log Analytics workspace
Get-AzResource -ResourceType "Microsoft.OperationalInsights/workspaces"
```

4. Copy the value for **ResourceId**.

To return results from a specific resource group, include the `-ResourceGroupName` parameter. For more information, see [Get-AzResource](#).

If you have more than one Automation account or workspace in the output of the preceding commands, you can find the name and other related properties that are part of the full resource ID of your Automation account by performing the following:

1. Sign in to the [Azure portal](#).

2. In the Azure portal, select your Automation account from the **Automation Accounts** page.
3. On the page of the selected Automation account, under **Account Settings**, select **Properties**.
4. In the **Properties** page, note the details shown below.

The screenshot shows the 'Properties' page for the 'MAIC-AA-Pri' Automation Account. The left sidebar lists 'Variables', 'Related Resources' (Linked workspace, Event grid, Start/Stop VM), and 'Account Settings' (Properties, Keys, Pricing, Source control, Run as accounts). The main pane shows the following details:

- Name:** MAIC-AA-Pri
- Subscription:** Tailwind Traders
- Subscription ID:** 51146ee0-c17b-44ba-b65a-1e3a40cdec48
- Created:** 11/7/2019, 4:03 PM
- Last modified:** 5/21/2020, 10:11 AM
- Resource group:** MAIC-RG
- Region:** eastus2

Configure diagnostic settings

Automation diagnostic settings supports forwarding the following platform logs and metric data:

- JobLogs
- JobStreams
- DSCNodeStatus
- Metrics - Total Jobs, Total Update Deployment Machine Runs, Total Update Deployment Runs

To start sending your Automation logs to Azure Monitor logs, review [create diagnostic settings](#) to understand the feature and methods available to configure diagnostic settings to send platform logs.

Azure Monitor log records

Azure Automation diagnostics create two types of records in Azure Monitor logs, tagged as `AzureDiagnostics`. The tables in the next sections are examples of records that Azure Automation generates and the data types that appear in log search results.

Job logs

PROPERTY	DESCRIPTION
TimeGenerated	Date and time when the runbook job executed.
RunbookName_s	The name of the runbook.

PROPERTY	DESCRIPTION
Caller_s	The caller that initiated the operation. Possible values are either an email address or system for scheduled jobs.
Tenant_g	GUID that identifies the tenant for the caller.
JobId_g	GUID that identifies the runbook job.
ResultType	The status of the runbook job. Possible values are: - New - Created - Started - Stopped - Suspended - Failed - Completed
Category	Classification of the type of data. For Automation, the value is JobLogs.
OperationName	The type of operation performed in Azure. For Automation, the value is Job.
Resource	The name of the Automation account
SourceSystem	System that Azure Monitor logs use to collect the data. The value is always Azure for Azure diagnostics.
ResultDescription	The runbook job result state. Possible values are: - Job is started - Job Failed - Job Completed
CorrelationId	The correlation GUID of the runbook job.
ResourceId	The Azure Automation account resource ID of the runbook.
SubscriptionId	The Azure subscription GUID for the Automation account.
ResourceGroup	The name of the resource group for the Automation account.
ResourceProvider	The resource provider. The value is MICROSOFT.AUTOMATION.
ResourceType	The resource type. The value is AUTOMATIONACCOUNTS.

Job streams

PROPERTY	DESCRIPTION
TimeGenerated	Date and time when the runbook job executed.
RunbookName_s	The name of the runbook.

PROPERTY	DESCRIPTION
Caller_s	The caller that initiated the operation. Possible values are either an email address or system for scheduled jobs.
StreamType_s	The type of job stream. Possible values are: -Progress - Output - Warning - Error - Debug - Verbose
Tenant_g	GUID that identifies the tenant for the caller.
JobId_g	GUID that identifies the runbook job.
ResultType	The status of the runbook job. Possible values are: - In Progress
Category	Classification of the type of data. For Automation, the value is JobStreams.
OperationName	Type of operation performed in Azure. For Automation, the value is Job.
Resource	The name of the Automation account.
SourceSystem	System that Azure Monitor logs use to collect the data. The value is always Azure for Azure diagnostics.
ResultDescription	Description that includes the output stream from the runbook.
CorrelationId	The correlation GUID of the runbook job.
ResourceId	The Azure Automation account resource ID of the runbook.
SubscriptionId	The Azure subscription GUID for the Automation account.
ResourceGroup	The name of the resource group for the Automation account.
ResourceProvider	The resource provider. The value is MICROSOFT.AUTOMATION.
ResourceType	The resource type. The value is AUTOMATIONACCOUNTS.

View Automation logs in Azure Monitor logs

Now that you started sending your Automation job streams and logs to Azure Monitor logs, let's see what you can do with these logs inside Azure Monitor logs.

To see the logs, run the following query: `AzureDiagnostics | where ResourceProvider == "MICROSOFT.AUTOMATION"`

Send an email when a runbook job fails or suspends

The following steps show how to set up alerts in Azure Monitor to notify you when something goes wrong with a runbook job.

To create an alert rule, start by creating a log search for the runbook job records that should invoke the alert. Click the **Alert** button to create and configure the alert rule.

1. From the Log Analytics workspace Overview page, click **View logs**.

2. Create a log search query for your alert by typing the following search into the query field:

```
AzureDiagnostics | where ResourceProvider == "MICROSOFT.AUTOMATION" and Category == "JobLogs" and  
(ResultType == "Failed" or ResultType == "Suspended")
```

You can also group by the runbook name by using:

```
AzureDiagnostics | where ResourceProvider == "MICROSOFT.AUTOMATION" and Category == "JobLogs" and  
(ResultType == "Failed" or ResultType == "Suspended") | summarize AggregatedValue = count() by  
RunbookName_s
```

If you set up logs from more than one Automation account or subscription to your workspace, you can group your alerts by subscription and Automation account. Automation account name can be found in the **Resource** field in the search of **JobLogs**.

3. To open the **Create rule** screen, click **New Alert Rule** at the top of the page. For more information on the options to configure the alert, see [Log alerts in Azure](#).

Find all jobs that have completed with errors

In addition to alerting on failures, you can find when a runbook job has a non-terminating error. In these cases, PowerShell produces an error stream, but the non-terminating errors don't cause your job to suspend or fail.

1. In your Log Analytics workspace, click **Logs**.

2. In the query field, type

```
AzureDiagnostics | where ResourceProvider == "MICROSOFT.AUTOMATION" and Category == "JobStreams" and  
StreamType_s == "Error" | summarize AggregatedValue = count() by JobId_g
```

3. Click the **Search** button.

View job streams for a job

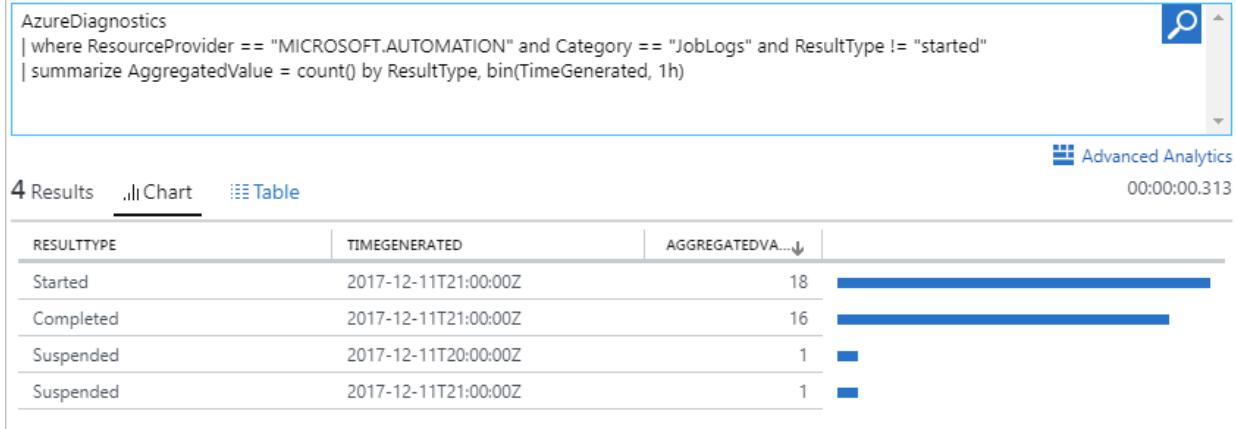
When you're debugging a job, you might also want to look into the job streams. The following query shows all the streams for a single job with GUID `2ebd22ea-e05e-4eb9-9d76-d73cbd4356e0`:

```
AzureDiagnostics  
| where ResourceProvider == "MICROSOFT.AUTOMATION" and Category == "JobStreams" and JobId_g == "2ebd22ea-  
e05e-4eb9-9d76-d73cbd4356e0"  
| sort by TimeGenerated asc  
| project ResultDescription
```

View historical job status

Finally, you might want to visualize your job history over time. You can use this query to search for the status of your jobs over time.

```
AzureDiagnostics  
| where ResourceProvider == "MICROSOFT.AUTOMATION" and Category == "JobLogs" and ResultType != "started"  
| summarize AggregatedValue = count() by ResultType, bin(TimeGenerated, 1h)
```



Filter job status output converted into a JSON object

Recently we changed the behavior of how the Automation log data is written to the `AzureDiagnostics` table in the Log Analytics service, where it no longer breaks down the JSON properties into separate fields. If you configured your runbook to format objects in the output stream in JSON format as separate columns, it is necessary to reconfigure your queries to parse that field to a JSON object in order to access those properties. This is accomplished using `parsejson` to access a specific JSON element in a known path.

For example, a runbook formats the `ResultDescription` property in the output stream in JSON format with multiple fields. To search for the status of your jobs that are in a failed state as specified in a field called `Status`, use this example query to search the `ResultDescription` with a status of `Failed`:

```
AzureDiagnostics  
| where Category == 'JobStreams'  
| extend jsonResourceDescription = parse_json(ResultDescription)  
| where jsonResourceDescription.Status == 'Failed'
```

Results | Chart | Columns | Display time (UTC+00:00) | Group columns

Completed. Showing results from the last 3 days.

TimeGenerated [UTC]	jsonResourceDescription	ResourceId	Category	ResourceGroup	SubscriptionId
ResourceProvider	MICROSOFTAUTOMATION				
Resource	MAIC-AA-PRI				
ResourceType	AUTOMATIONACCOUNTS				
OperationName	Job				
ResultType	In Progress				
CorrelationId	edfa026-da57-4b70-9218-8ec025c9aaa5				
> ResultDescription	{ "JobId": "e092fb7a-6f6e-4bd0-ab20-ff467766600d", "RanOn": "myhybridworkergroupname", "TenantId": "12345678-abcd-1234-5678-1234567890ab", "Status": "Failed", "JobInput": { "ModuleName": "TestRunbook", "Parameters": {} } }				
Tenant_g	7a6d0eca-222b-49cf-8415-f24adabdee37				
JobId_g	9db8a42b-76d4-48a0-b949-2ee8b53682cd				
RunbookName_s	POSH-JSONOutput-Test				
StreamType_s	Output				
SourceSystem	Azure				
Type	AzureDiagnostics				
_ResourceId	/subscriptions/68627f8c-65b8-4601-b48e-b032a81f8cf0/resourcegroups/maic-rg/providers/microsoft.automation/automationaccounts/maic-aa-pri				
> jsonResourceDescription	{ "JobId": "e092fb7a-6f6e-4bd0-ab20-ff467766600d", "RanOn": "myhybridworkergroupname", "TenantId": "12345678-abcd-1234-5678-1234567890ab", "Status": "Failed", "JobInput": { "ModuleName": "TestRunbook", "Parameters": {} } }				

Next steps

- To learn how to construct search queries and review the Automation job logs with Azure Monitor logs, see [Log searches in Azure Monitor logs](#).
- To understand creation and retrieval of output and error messages from runbooks, see [Monitor runbook output](#).
- To learn more about runbook execution, how to monitor runbook jobs, and other technical details, see [Runbook execution in Azure Automation](#).

- To learn more about Azure Monitor logs and data collection sources, see [Collecting Azure storage data in Azure Monitor logs overview](#).
- For help troubleshooting Log Analytics, see [Troubleshooting why Log Analytics is no longer collecting data](#).

Troubleshoot runbook issues

9/10/2021 • 21 minutes to read • [Edit Online](#)

This article describes runbook issues that might occur and how to resolve them. For general information, see [Runbook execution in Azure Automation](#).

Diagnose runbook issues

When you receive errors during runbook execution in Azure Automation, you can use the following steps to help diagnose the issues:

1. Ensure that your runbook script executes successfully on your local machine.

For language reference and learning modules, see the [PowerShell Docs](#) or [Python Docs](#). Running your script locally can discover and resolve common errors, such as:

- Missing modules
- Syntax errors
- Logic errors

2. Investigate runbook [error streams](#).

Look at these streams for specific messages, and compare them to the errors documented in this article.

3. Ensure that your nodes and Automation workspace have the required modules.

If your runbook imports any modules, verify that they're available to your Automation account by using the steps in [Import modules](#). Update your PowerShell modules to the latest version by following the instructions in [Update Azure PowerShell modules in Azure Automation](#). For more troubleshooting information, see [Troubleshoot modules](#).

4. If your runbook is suspended or unexpectedly fails:

- [Renew the certificate](#) if the Run As account has expired.
- [Renew the webhook](#) if you're trying to use an expired webhook to start the runbook.
- [Check job statuses](#) to determine current runbook statuses and some possible causes of the issue.
- [Add additional output](#) to the runbook to identify what happens before the runbook is suspended.
- [Handle any exceptions](#) that are thrown by your job.

5. Do this step if the runbook job or the environment on Hybrid Runbook Worker doesn't respond.

If you're running your runbooks on a Hybrid Runbook Worker instead of in Azure Automation, you might need to [troubleshoot the hybrid worker itself](#).

Scenario: Access blocked to Azure Storage, or Azure Key Vault, or Azure SQL

This scenario uses [Azure Storage](#) as an example; however, the information is equally applicable to [Azure Key Vault](#) and [Azure SQL](#).

Issue

Attempting to access Azure Storage from a Runbook results in an error similar to the following message:

The remote server returned an error: (403) Forbidden. HTTP Status Code: 403 - HTTP Error Message: This request is not authorized to perform this operation.

Cause

The Azure Firewall on Azure Storage is enabled.

Resolution

Enabling the Azure Firewall on [Azure Storage](#), [Azure Key Vault](#), or [Azure SQL](#) blocks access from Azure Automation runbooks for those services. Access will be blocked even when the firewall exception to allow trusted Microsoft services is enabled, as Automation is not a part of the trusted services list. With an enabled firewall, access can only be made by using a Hybrid Runbook Worker and a [virtual network service endpoint](#).

Scenario: Runbook fails with a No permission or Forbidden 403 error

Issue

Your runbook fails with a No permission or Forbidden 403 error, or equivalent.

Cause

Run As accounts might not have the same permissions against Azure resources as your current Automation account.

Resolution

Ensure that your Run As account has [permissions to access any resources](#) used in your script.

Scenario: Sign-in to Azure account failed

Issue

You receive one of the following errors when you work with the `Connect-AzAccount` cmdlet:

Unknown_user_type: Unknown User Type

No certificate was found in the certificate store with thumbprint

Cause

These errors occur if the credential asset name isn't valid. They might also occur if the user name and password that you used to set up the Automation credential asset aren't valid.

Resolution

To determine what's wrong, follow these steps:

1. Make sure that you don't have any special characters. These characters include the `\@` character in the Automation credential asset name that you're using to connect to Azure.
2. Check to see if you can use the user name and password that are stored in the Azure Automation credential in your local PowerShell ISE editor. Run the following cmdlets in the PowerShell ISE.

```
$Cred = Get-Credential  
#Using Azure Service Management  
Add-AzureAccount -Credential $Cred  
#Using Azure Resource Manager  
Connect-AzAccount -Credential $Cred
```

3. If your authentication fails locally, you haven't set up your Azure Active Directory (Azure AD) credentials properly. To get the Azure AD account set up correctly, see the article [Authenticate to Azure using Azure Active Directory](#).

4. If the error appears to be transient, try adding retry logic to your authentication routine to make authenticating more robust.

```
# Get the connection "AzureRunAsConnection"
$connectionName = "AzureRunAsConnection"
$servicePrincipalConnection = Get-AutomationConnection -Name $connectionName

$logonAttempt = 0
$logonResult = $False

while(!($connectionResult) -And ($logonAttempt -le 10))
{
    $LogonAttempt++
    #Logging in to Azure...
    $connectionResult = Connect-AzAccount ` 
        -ServicePrincipal ` 
        -Tenant $servicePrincipalConnection.TenantId ` 
        -ApplicationId $servicePrincipalConnection.ApplicationId ` 
        -CertificateThumbprint $servicePrincipalConnection.CertificateThumbprint

    Start-Sleep -Seconds 30
}
```

Scenario: Run Login-AzureRMAccount to log in

Issue

You receive the following error when you run a runbook:

```
Run Login-AzureRMAccount to login.
```

Cause

This error can occur when you're not using a Run As account or the Run As account has expired. For more information, see [Azure Automation Run As accounts overview](#).

This error has two primary causes:

- There are different versions of the AzureRM or Az module.
- You're trying to access resources in a separate subscription.

Resolution

If you receive this error after you update one AzureRM or Az module, update all your modules to the same version.

If you're trying to access resources in another subscription, follow these steps to configure permissions:

1. Go to the Automation Run As account, and copy the **Application ID** and **Thumbprint**.

Azure Run As Account

Properties

Renew certificate Delete

Azure Active Directory Application

Display Name [REDACTED]

Certificate

Name AzureRunAsCertificate
Expiration 3/12/2020, 6:00 PM
Thumbprint [REDACTED]

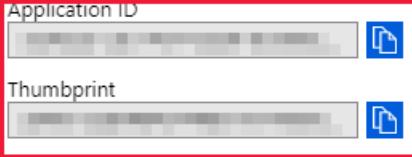
Connection

Name AzureRunAsConnection

Application ID [REDACTED]
Thumbprint [REDACTED]
Subscription ID [REDACTED]
Tenant ID 72f988bf-86f1-41af-91ab-2d7cd011...

Service Principal

Service Principal Object ID 00c2d42c-ed25-49fa-9f6f-2f485695244b



2. Go to the subscription's **Access control (IAM)** where the Automation account is *not* hosted, and add a new role assignment.

- Access control (IAM)

Subscription

Search (Ctrl+/
Add Edit columns Refresh Remove

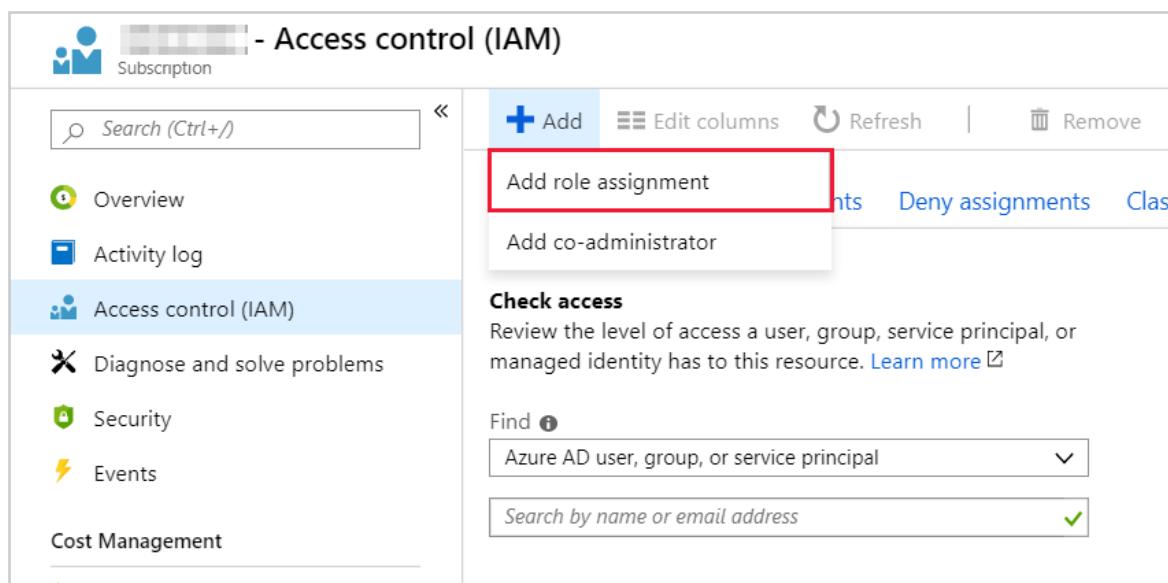
Add role assignment 

Add co-administrator

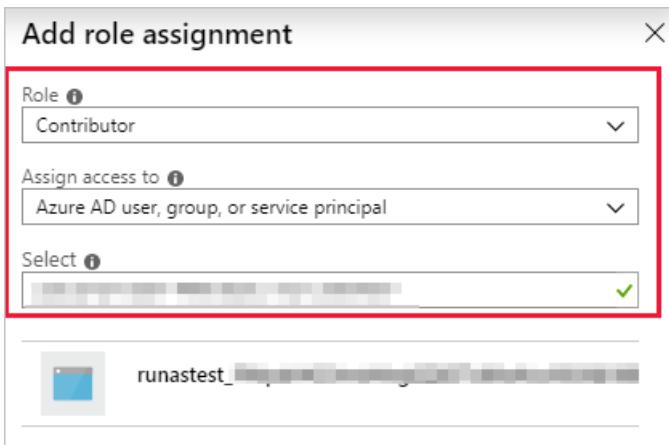
Check access
Review the level of access a user, group, service principal, or managed identity has to this resource. [Learn more](#)

Find Azure AD user, group, or service principal

Search by name or email address



3. Add the Application ID collected earlier. Select **Contributor** permissions.



4. Copy the name of the subscription.

5. You can now use the following runbook code to test the permissions from your Automation account to the other subscription. Replace <CertificateThumbprint> with the value copied in step 1. Replace "<SubscriptionName>" with the value copied in step 4.

```
$Conn = Get-AutomationConnection -Name AzureRunAsConnection
Connect-AzAccount -ServicePrincipal -Tenant $Conn.TenantID -ApplicationId $Conn.ApplicationID -
CertificateThumbprint "<CertificateThumbprint>"
#Select the subscription you want to work with
Select-AzSubscription -SubscriptionName '<YourSubscriptionNameGoesHere>'

#Test and get outputs of the subscriptions you granted access.
$subscriptions = Get-AzSubscription
foreach($subscription in $subscriptions)
{
    Set-AzContext $subscription
    Write-Output $subscription.Name
}
```

Scenario: Unable to find the Azure subscription

Issue

You receive the following error when you work with the `Select-AzureSubscription`, `Select-AzureRMSubscription`, or `Select-AzSubscription` cmdlet:

```
The subscription named <subscription name> cannot be found.
```

Error

This error can occur if:

- The subscription name isn't valid.
- The Azure AD user who's trying to get the subscription details isn't configured as an administrator of the subscription.
- The cmdlet isn't available.

Resolution

Follow these steps to determine if you've authenticated to Azure and have access to the subscription that you're trying to select:

1. To make sure that your script works standalone, test it outside of Azure Automation.
2. Make sure that your script runs the `Connect-AzAccount` cmdlet before running the `Select-*` cmdlet.

3. Add `Disable-AzContextAutosave -Scope Process` to the beginning of your runbook. This cmdlet ensures that any credentials apply only to the execution of the current runbook.
4. If you still see the error message, modify your code by adding the `AzContext` parameter for `Connect-AzAccount`, and then execute the code.

```
Disable-AzContextAutosave -Scope Process

$Conn = Get-AutomationConnection -Name AzureRunAsConnection
Connect-AzAccount -ServicePrincipal -Tenant $Conn.TenantID -ApplicationId $Conn.ApplicationID -
CertificateThumbprint $Conn.CertificateThumbprint

 = Get-AzContext

Get-AzVM -ResourceGroupName myResourceGroup -AzContext $context
```

Scenario: Runbooks fail when dealing with multiple subscriptions

Issue

When executing runbooks, the runbook fails to manage Azure resources.

Cause

The runbook isn't using the correct context when running. This may be because the runbook is accidentally trying to access the incorrect subscription.

You may see errors like this one:

```
Get-AzVM : The client '<automation-runas-account-guid>' with object id '<automation-runas-account-guid>' does not have authorization to perform action 'Microsoft.Compute/virtualMachines/read' over scope '/subscriptions/<subscriptionIdOfSubscriptionWhichDoesntContainTheVM>/resourceGroups/REsourceGroupName/providers/Microsoft.Compute/virtualMachines/VMName'.
 ErrorCode: AuthorizationFailed
 StatusCode: 403
 ReasonPhrase: Forbidden Operation
 ID : <AGuidRepresentingTheOperation> At line:51 char:7 + $vm = Get-AzVM -ResourceGroupName
 $ResourceGroupName -Name $UNBV... +
```

Resolution

The subscription context might be lost when a runbook invokes multiple runbooks. To avoid accidentally trying to access the incorrect subscription you should follow the guidance below.

- To avoid referencing the wrong subscription, disable context saving in your Automation runbooks by using the following code at the start of each runbook.

```
Disable-AzContextAutosave -Scope Process
```

- The Azure PowerShell cmdlets support the `-DefaultProfile` parameter. This was added to all Az and AzureRm cmdlets to support running multiple PowerShell scripts in the same process, allowing you to specify the context and which subscription to use for each cmdlet. With your runbooks, you should save the context object in your runbook when the runbook is created (that is, when an account signs in) and every time it's changed, and reference the context when you specify an Az cmdlet.

NOTE

You should pass in a context object even when manipulating the context directly using cmdlets such as [Set-AzContext](#) or [Select-AzSubscription](#).

```
$servicePrincipalConnection=Get-AutomationConnection -Name $connectionName  
$context = Add-AzAccount `  
    -ServicePrincipal `  
    -TenantId $servicePrincipalConnection.TenantId `  
    -ApplicationId $servicePrincipalConnection.ApplicationId `  
    -Subscription 'cd4dxxxx-xxxx-xxxx-xxxx-xxxxxxxxx9749' `  
    -CertificateThumbprint $servicePrincipalConnection.CertificateThumbprint  
$context = Set-AzContext -SubscriptionName $subscription `  
    -DefaultProfile $context  
Get-AzVm -DefaultProfile $context
```

Scenario: Authentication to Azure fails because multifactor authentication is enabled

Issue

You receive the following error when authenticating to Azure with your Azure user name and password:

```
Add-AzureAccount: AADSTS50079: Strong authentication enrollment (proof-up) is required
```

Cause

If you have multifactor authentication on your Azure account, you can't use an Azure Active Directory user to authenticate to Azure. Instead, you need to use a certificate or a service principal to authenticate.

Resolution

To use a Classic Run As account with Azure classic deployment model cmdlets, see [Create a Classic Run As account to manage Azure services](#). To use a service principal with Azure Resource Manager cmdlets, see [Creating service principal using Azure portal](#) and [Authenticating a service principal with Azure Resource Manager](#).

Scenario: Runbook fails with "A task was canceled" error message

Issue

Your runbook fails with an error similar to the following example:

```
Exception: A task was canceled.
```

Cause

This error can be caused by using outdated Azure modules.

Resolution

You can resolve this error by updating your Azure modules to the latest version:

1. In your Automation account, select **Modules**, and then select **Update Azure modules**.
2. The update takes roughly 15 minutes. After it's finished, rerun the runbook that failed.

To learn more about updating your modules, see [Update Azure modules in Azure Automation](#).

Scenario: Term not recognized as the name of a cmdlet, function, or script

Issue

Your runbook fails with an error similar to the following example:

```
The term 'Connect-AzAccount' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling of the name, or if the path was included verify that the path is correct and try again.
```

Cause

This error can happen for the following reasons:

- The module that contains the cmdlet isn't imported into the Automation account.
- The module that contains the cmdlet is imported but is out of date.

Resolution

Do one of the following tasks to resolve this error:

- For an Azure module, see [How to update Azure PowerShell modules in Azure Automation](#) to learn how to update your modules in your Automation account.
- For a non-Azure module, make sure that the module is imported into your Automation account.

Scenario: Cmdlet fails in PnP PowerShell runbook on Azure Automation

Issue

When a runbook writes a PnP PowerShell-generated object to the Azure Automation output directly, cmdlet output can't stream back to Automation.

Cause

This issue most commonly occurs when Azure Automation processes runbooks that invoke PnP PowerShell cmdlets, for example, `add-pnplistitem`, without catching the return objects.

Resolution

Edit your scripts to assign any return values to variables so that the cmdlets don't attempt to write whole objects to the standard output. A script can redirect the output stream to a cmdlet, as shown here.

```
$null = add-pnplistitem
```

If your script parses cmdlet output, the script must store the output in a variable and manipulate the variable instead of simply streaming the output.

```
$SomeVariable = add-pnplistitem ....  
if ($SomeVariable.someproperty -eq ....
```

Scenario: Cmdlet not recognized when executing a runbook

Issue

Your runbook job fails with the error:

```
<cmdlet name>: The term <cmdlet name> is not recognized as the name of a cmdlet, function, script file, or operable program.
```

Cause

This error is caused when the PowerShell engine can't find the cmdlet you're using in your runbook. It's possible that the module containing the cmdlet is missing from the account, there's a name conflict with a runbook name, or the cmdlet also exists in another module and Automation can't resolve the name.

Resolution

Use any of the following solutions to fix the problem:

- Make sure that you've entered the cmdlet name correctly.
- Ensure that the cmdlet exists in your Automation account and that there are no conflicts. To verify if the cmdlet is present, open a runbook in edit mode and search for the cmdlet you want to find in the library, or run `Get-Command <CommandName>`. After you've validated that the cmdlet is available to the account, and that there are no name conflicts with other cmdlets or runbooks, add the cmdlet to the canvas. Make sure that you're using a valid parameter set in your runbook.
- If you do have a name conflict and the cmdlet is available in two different modules, resolve the issue by using the fully qualified name for the cmdlet. For example, you can use `ModuleName\CmdletName`.
- If you're executing the runbook on-premises in a hybrid worker group, ensure that the module and cmdlet are installed on the machine that hosts the hybrid worker.

Scenario: Incorrect object reference on call to Add-AzAccount

Issue

You receive this error when you work with `Add-AzAccount`, which is an alias for the `Connect-AzAccount` cmdlet:

```
Add-AzAccount : Object reference not set to an instance of an object
```

Cause

This error can occur if the runbook doesn't do the proper steps before calling `Add-AzAccount` to add the Automation account. An example of one of the necessary steps is signing in with a Run As account. For the correct operations to use in your runbook, see [Runbook execution in Azure Automation](#).

Scenario: Object reference not set to an instance of an object

Issue

You receive the following error when you invoke a child runbook with the `Wait` parameter and the Output stream contains an object:

```
Object reference not set to an instance of an object
```

Cause

If the stream contains objects, `Start-AzAutomationRunbook` doesn't handle the Output stream correctly.

Resolution

Implement a polling logic, and use the `Get-AzAutomationJobOutput` cmdlet to retrieve the output. A sample of this logic is defined here:

```

$automationAccountName = "ContosoAutomationAccount"
$runbookName = "ChildRunbookExample"
$resourceGroupName = "ContosoRG"

function IsJobTerminalState([string] $status) {
    return $status -eq "Completed" -or $status -eq "Failed" -or $status -eq "Stopped" -or $status -eq "Suspended"
}

$job = Start-AzAutomationRunbook -AutomationAccountName $automationAccountName -Name $runbookName -
ResourceGroupName $resourceGroupName
$pollingSeconds = 5
$maxTimeout = 10800
$waitTime = 0
while($false -eq (IsJobTerminalState $job.Status) -and $waitTime -lt $maxTimeout) {
    Start-Sleep -Seconds $pollingSeconds
    $waitTime += $pollingSeconds
    $job = $job | Get-AzAutomationJob
}

$job | Get-AzAutomationJobOutput | Get-AzAutomationJobOutputRecord | Select-Object -ExpandProperty Value

```

Scenario: Runbook fails because of deserialized object

Issue

Your runbook fails with the error:

```

Cannot bind parameter <ParameterName>.

Cannot convert the <ParameterType> value of type Deserialized <ParameterType> to type <ParameterType>.

```

Cause

If your runbook is a PowerShell Workflow, it stores complex objects in a deserialized format to persist your runbook state if the workflow is suspended.

Resolution

Use any of the following solutions to fix this problem:

- If you're piping complex objects from one cmdlet to another, wrap these cmdlets in an `InlineScript` activity.
- Pass the name or value that you need from the complex object instead of passing the entire object.
- Use a PowerShell runbook instead of a PowerShell Workflow runbook.

Scenario: 400 Bad Request status when calling a webhook

Issue

When you try to invoke a webhook for an Azure Automation runbook, you receive the following error:

```
400 Bad Request : This webhook has expired or is disabled
```

Cause

The webhook that you're trying to call is either disabled or is expired.

Resolution

If the webhook is disabled, you can re-enable it through the Azure portal. If the webhook has expired, you must delete and then re-create it. You can only [renew a webhook](#) if it hasn't already expired.

Scenario: 429: The request rate is currently too large

Issue

You receive the following error message when running the `Get-AzAutomationJobOutput` cmdlet:

```
429: The request rate is currently too large. Please try again
```

Cause

This error can occur when retrieving job output from a runbook that has many [verbose streams](#).

Resolution

Do one of the following to resolve this error:

- Edit the runbook, and reduce the number of job streams that it emits.
- Reduce the number of streams to be retrieved when running the cmdlet. To do this, you can set the value of the `Stream` parameter for the [Get-AzAutomationJobOutput](#) cmdlet to retrieve only Output streams.

Scenario: Runbook job fails because allocated quota was exceeded

Issue

Your runbook job fails with the error:

```
The quota for the monthly total job run time has been reached for this subscription
```

Cause

This error occurs when the job execution exceeds the 500-minute free quota for your account. This quota applies to all types of job execution tasks. Some of these tasks are testing a job, starting a job from the portal, executing a job by using webhooks, or scheduling a job to execute using either the Azure portal or your datacenter. To learn more about pricing for Automation, see [Automation pricing](#).

Resolution

If you want to use more than 500 minutes of processing per month, change your subscription from the Free tier to the Basic tier:

1. Sign in to your Azure subscription.
2. Select the Automation account to upgrade.
3. Select **Settings**, and then select **Pricing**.
4. Select **Enable** on the page bottom to upgrade your account to the Basic tier.

Scenario: Runbook output stream greater than 1 MB

Issue

Your runbook running in the Azure sandbox fails with the following error:

```
The runbook job failed due to a job stream being larger than 1MB, this is the limit supported by an Azure Automation sandbox.
```

Cause

This error occurs because your runbook attempted to write too much exception data to the output stream.

Resolution

There's a 1 MB limit on the job output stream. Ensure that your runbook encloses calls to an executable or subprocess by using `try` and `catch` blocks. If the operations throw an exception, have the code write the message from the exception into an Automation variable. This technique prevents the message from being written into the job output stream. For Hybrid Runbook Worker jobs executed, the output stream truncated to 1 MB is displayed with no error message.

Scenario: Runbook job start attempted three times, but fails to start each time

Issue

Your runbook fails with the following error:

```
The job was tried three times but it failed
```

Cause

This error occurs because of one of the following issues:

- **Memory limit.** A job might fail if it's using more than 400 MB of memory. The documented limits on memory allocated to a sandbox are found at [Automation service limits](#).
- **Network sockets.** Azure sandboxes are limited to 1,000 concurrent network sockets. For more information, see [Automation service limits](#).
- **Module incompatible.** Module dependencies might not be correct. In this case, your runbook typically returns a `Command not found` or `Cannot bind parameter` message.
- **No authentication with Active Directory for sandbox.** Your runbook attempted to call an executable or subprocess that runs in an Azure sandbox. Configuring runbooks to authenticate with Azure AD by using the Azure Active Directory Authentication Library (ADAL) isn't supported.

Resolution

- **Memory limit, network sockets.** Suggested ways to work within the memory limits are to split the workload among multiple runbooks, process less data in memory, avoid writing unnecessary output from your runbooks, and consider how many checkpoints are written into your PowerShell workflow runbooks. Use the clear method, such as `$myVar.clear`, to clear out variables and use `[GC]::collect` to run garbage collection immediately. These actions reduce the memory footprint of your runbook during runtime.
- **Module incompatible.** Update your Azure modules by following the steps in [How to update Azure PowerShell modules in Azure Automation](#).
- **No authentication with Active Directory for sandbox.** When you authenticate to Azure AD with a runbook, ensure that the Azure AD module is available in your Automation account. Be sure to grant the Run As account the necessary permissions to perform the tasks that the runbook automates.

If your runbook can't call an executable or subprocess running in an Azure sandbox, use the runbook on a [Hybrid Runbook Worker](#). Hybrid workers aren't limited by the memory and network limits that Azure sandboxes have.

Scenario: PowerShell job fails with "Cannot invoke method" error message

Issue

You receive the following error message when you start a PowerShell job in a runbook that runs in Azure:

Exception was thrown - Cannot invoke method. Method invocation is supported only on core types in this language mode.

Cause

This error might indicate that runbooks that run in an Azure sandbox can't run in the [Full Language mode](#).

Resolution

There are two ways to resolve this error:

- Instead of using [Start-Job](#), use [Start-AzAutomationRunbook](#) to start the runbook.
- Try running the runbook on a Hybrid Runbook Worker.

To learn more about this behavior and other behaviors of Azure Automation runbooks, see [Runbook execution in Azure Automation](#).

Scenario: A long-running runbook fails to complete

Issue

Your runbook shows in a Stopped state after running for three hours. You might also receive this error:

The job was evicted and subsequently reached a Stopped state. The job cannot continue running.

This behavior is by design in Azure sandboxes because of the [fair share](#) monitoring of processes within Azure Automation. If a process executes longer than three hours, fair share automatically stops a runbook. The status of a runbook that goes past the fair share time limit differs by runbook type. PowerShell and Python runbooks are set to a Stopped status. PowerShell Workflow runbooks are set to Failed.

Cause

The runbook ran over the three-hour limit allowed by fair share in an Azure sandbox.

Resolution

One recommended solution is to run the runbook on a [Hybrid Runbook Worker](#). Hybrid workers aren't limited by the three-hour fair share runbook limit that Azure sandboxes have. Runbooks that run on Hybrid Runbook Workers should be developed to support restart behaviors if there are unexpected local infrastructure issues.

Another solution is to optimize the runbook by creating [child runbooks](#). If your runbook loops through the same function on several resources, for example, in a database operation on several databases, you can move the function to a child runbook. Each child runbook executes in parallel in a separate process. This behavior decreases the total amount of time for the parent runbook to complete.

The PowerShell cmdlets that enable the child runbook scenario are:

- [Start-AzAutomationRunbook](#). This cmdlet allows you to start a runbook and pass parameters to the runbook.
- [Get-AzAutomationJob](#). If there are operations that need to be performed after the child runbook completes, this cmdlet allows you to check the job status for each child.

Scenario: Error in job streams about the get_SerializationSettings method

Issue

You see the following error in your job streams for a runbook:

```
Connect-AzAccount : Method 'get_SerializationSettings' in type
'Microsoft.Azure.Management.Internal.Resources.ResourceManagementClient' from assembly
'Microsoft.Azure.Commands.ResourceManager.Common, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35'
does not have an implementation.
At line:16 char:1
+ Connect-AzAccount -ServicePrincipal -Tenant $Conn.TenantID -Appl ...
+ ~~~~~
+ CategoryInfo          : NotSpecified: (:) [Connect-AzAccount], TypeLoadException
+ FullyQualifiedErrorId :
System.TypeLoadException,Microsoft.Azure.Commands.Profile.ConnectAzAccountCommand
```

Cause

This error is probably caused by using an incomplete migration from AzureRM to Az modules in your runbook. This situation can cause Azure Automation to start a runbook job by using only AzureRM modules, and then start another job by using only Az modules, which leads to a sandbox crash.

Resolution

We don't recommend the use of Az and AzureRM cmdlets in the same runbook. To learn more about the correct use of the modules, see [Migrate to Az modules](#).

Scenario: Access denied when using Azure sandbox for runbook or application

Issue

When your runbook or application attempts to run in an Azure sandbox, the environment denies access.

Cause

This issue can occur because Azure sandboxes prevent access to all out-of-process COM servers. For example, a sandboxed application or runbook can't call into Windows Management Instrumentation (WMI) or into the Windows Installer service (msiserver.exe).

Resolution

For details about the use of Azure sandboxes, see [Runbook execution environment](#).

Scenario: Invalid Forbidden status code when using Key Vault inside a runbook

Issue

When you try to access Azure Key Vault through an Azure Automation runbook, you get the following error:

```
Operation returned an invalid status code 'Forbidden'
```

Cause

Possible causes for this issue are:

- Not using a Run As account.
- Insufficient permissions.

Resolution

Not using a Run As account

Follow [Step 5 - Add authentication to manage Azure resources](#) to ensure that you are using a Run As account to access Key Vault.

Insufficient permissions

Add [permissions to Key Vault](#) to ensure that your Run As account has sufficient permissions to access Key Vault.

Recommended documents

- [Runbook execution in Azure Automation](#)
- [Starting a runbook in Azure Automation](#)

Next steps

If you don't see your problem here or you're unable to resolve your issue, try one of the following channels for more support:

- Get answers from Azure experts through [Azure Forums](#).
- Connect with [@AzureSupport](#), the official Microsoft Azure account for improving customer experience. Azure Support connects you to the Azure community for answers, support, and experts.
- If you need more help, you can file an Azure support incident. Go to the [Azure support site](#), and select **Get Support**.

Data to collect when opening a case for Microsoft Azure Automation

3/5/2021 • 2 minutes to read • [Edit Online](#)

This article describes some of the information that you should gather before you open a case for Azure Automation with Microsoft Azure Support. This information is not required to open the case. However, it can help Microsoft resolve your problem more quickly. Also, you may be asked for this data by the support engineer after you open the case.

Basic data

Collect the basic data described in the Knowledge Base article [4034605 - How to capture Azure Automation-scripted diagnostics](#).

Data for Update Management issues on Linux

1. In addition to the items that are listed in KB [4034605](#), run the following log collection tool:

[OMS Linux Agent Log Collector](#)

2. Compress the contents of the `/var/opt/microsoft/omsagent/run/automationworker/` folder, then send the compressed file to Azure Support.
3. Verify that the ID for the workspace that the Log Analytics agent for Linux reports to is the same as the ID for the workspace being monitored for updates.

Data for Update Management issues on Windows

1. Collect data for the items listed in [4034605](#).
2. Export the following event logs into the EVTX format:
 - System
 - Application
 - Security
 - Operations Manager
 - Microsoft-SMA/Operational
3. Verify that the ID of the workspace that the agent reports to is the same as the ID for the workspace being monitored by Windows Updates.

Data for job issues

1. Collect data for the items listed in [4034605](#).
2. Collect the job ID for the job that has an issue:
 - a. In the Azure portal, go to **Automation Accounts**.
 - b. Select the Automation account that you are troubleshooting, and note the name.
 - c. Select **Jobs**.
 - d. Choose the job that you are troubleshooting.

e. In the Job Summary pane, look for the GUID value in Job ID.

STATUS	RUNBOOK	CREATED	LAST UPDATED
Completed	AdoyleTestingrb	6/24/2016 3:26 PM	6/29/2016 3:26 PM
Completed	AdoyleTestingrb	6/28/2016 3:26 PM	6/28/2016 3:26 PM
Completed	AdoyleTestingrb	6/27/2016 3:26 PM	6/27/2016 3:26 PM
Completed	AdoyleTestingrb	6/26/2016 3:26 PM	6/26/2016 3:26 PM
Completed	AdoyleTestingrb	6/25/2016 3:26 PM	6/25/2016 3:26 PM
Completed	AdoyleTestingrb	6/24/2016 3:26 PM	6/24/2016 3:27 PM
Completed	AdoyleTestingrb	6/23/2016 3:26 PM	6/23/2016 3:27 PM
Completed	AdoyleTestingrb	6/22/2016 3:26 PM	6/22/2016 3:26 PM
Completed	AdoyleTestingrb	6/21/2016 3:26 PM	6/21/2016 3:26 PM
Completed	AdoyleTestingrb	6/20/2016 3:26 PM	6/20/2016 3:27 PM
Completed	AdoyleTestingrb	6/19/2016 3:26 PM	6/19/2016 3:27 PM
Completed	AdoyleTestingrb	6/18/2016 3:26 PM	6/18/2016 3:27 PM
Completed	AdoyleTestingrb	6/17/2016 3:26 PM	6/17/2016 3:26 PM
Completed	AdoyleTestingrb	6/16/2016 3:26 PM	6/16/2016 3:27 PM
Completed	AdoyleTestingrb	6/15/2016 3:26 PM	6/15/2016 3:26 PM
Completed	AdoyleTestingrb	6/14/2016 3:23 PM	6/14/2016 3:26 PM
Completed	AdoyleTestingrb	6/13/2016 3:23 PM	6/13/2016 3:27 PM
Completed	AdoyleTestingrb	6/12/2016 3:26 PM	6/12/2016 3:27 PM
Completed	AdoyleTestingrb	6/11/2016 3:26 PM	6/11/2016 3:26 PM

3. Collect a sample of the script that you are running.

4. Collect the log files:

- In the Azure portal, go to **Automation Accounts**.
- Select the Automation account that you are troubleshooting.
- Select **Jobs**.
- Choose the job that you are troubleshooting.
- Select **All Logs**.
- In the resulting pane, collect the data.

STATUS	RUNBOOK	CREATED	LAST UPDATED
Completed	AdoyleTestingrb	6/29/2016 3:26 PM	6/29/2016 3:26 PM
Completed	AdoyleTestingrb	6/28/2016 3:26 PM	6/28/2016 3:26 PM
Completed	AdoyleTestingrb	6/27/2016 3:26 PM	6/27/2016 3:26 PM
Completed	AdoyleTestingrb	6/26/2016 3:26 PM	6/26/2016 3:26 PM
Completed	AdoyleTestingrb	6/25/2016 3:26 PM	6/25/2016 3:26 PM
Completed	AdoyleTestingrb	6/24/2016 3:26 PM	6/24/2016 3:27 PM
Completed	AdoyleTestingrb	6/23/2016 3:26 PM	6/23/2016 3:27 PM
Completed	AdoyleTestingrb	6/22/2016 3:26 PM	6/22/2016 3:27 PM
Completed	AdoyleTestingrb	6/21/2016 3:26 PM	6/21/2016 3:26 PM
Completed	AdoyleTestingrb	6/20/2016 3:26 PM	6/21/2016 3:26 PM
Completed	AdoyleTestingrb	6/19/2016 3:26 PM	6/19/2016 3:27 PM
Completed	AdoyleTestingrb	6/18/2016 3:26 PM	6/18/2016 3:27 PM
Completed	AdoyleTestingrb	6/17/2016 3:26 PM	6/17/2016 3:26 PM
Completed	AdoyleTestingrb	6/16/2016 3:26 PM	6/16/2016 3:27 PM
Completed	AdoyleTestingrb	6/15/2016 3:26 PM	6/15/2016 3:26 PM
Completed	AdoyleTestingrb	6/14/2016 3:23 PM	6/14/2016 3:26 PM
Completed	AdoyleTestingrb	6/13/2016 3:23 PM	6/13/2016 3:27 PM
Completed	AdoyleTestingrb	6/12/2016 3:26 PM	6/12/2016 3:27 PM
Completed	AdoyleTestingrb	6/11/2016 3:26 PM	6/11/2016 3:26 PM

Data for module issues

In addition to the [basic data items](#), gather the following information:

- The steps you have followed, so that the problem can be reproduced.
- Screenshots of any error messages.
- Screenshots of the current modules and their version numbers.

Next steps

If you need more help:

- Get answers from Azure experts through [Azure Forums](#).
- Connect with [@AzureSupport](#), the official Microsoft Azure account for improving customer experience by connecting the Azure community to the right resources: answers, support, and experts.
- File an Azure support incident. Go to the [Azure support site](#) and select **Get Support**.

Deploy a Windows Hybrid Runbook Worker

8/24/2021 • 12 minutes to read • [Edit Online](#)

You can use the user Hybrid Runbook Worker feature of Azure Automation to run runbooks directly on an Azure or non-Azure machine, including servers registered with [Azure Arc-enabled servers](#). From the machine or server that's hosting the role, you can run runbooks directly against it and against resources in the environment to manage those local resources.

Azure Automation stores and manages runbooks and then delivers them to one or more designated machines. This article describes how to deploy a user Hybrid Runbook Worker on a Windows machine, how to remove the worker, and how to remove a Hybrid Runbook Worker group.

After you successfully deploy a runbook worker, review [Run runbooks on a Hybrid Runbook Worker](#) to learn how to configure your runbooks to automate processes in your on-premises datacenter or other cloud environment.

Prerequisites

Before you start, make sure that you have the following.

A Log Analytics workspace

The Hybrid Runbook Worker role depends on an Azure Monitor Log Analytics workspace to install and configure the role. You can create it through [Azure Resource Manager](#), through [PowerShell](#), or in the [Azure portal](#).

If you don't have an Azure Monitor Log Analytics workspace, review the [Azure Monitor Log design guidance](#) before you create the workspace.

Log Analytics agent

The Hybrid Runbook Worker role requires the [Log Analytics agent](#) for the supported Windows operating system. For servers or machines hosted outside of Azure, you can install the Log Analytics agent using [Azure Arc-enabled servers](#).

Supported Windows operating system

The Hybrid Runbook Worker feature supports the following operating systems:

- Windows Server 2019 (including Server Core)
- Windows Server 2016, version 1709 and 1803 (excluding Server Core)
- Windows Server 2012, 2012 R2
- Windows Server 2008 SP2 (x64), 2008 R2
- Windows 10 Enterprise (including multi-session) and Pro
- Windows 8 Enterprise and Pro
- Windows 7 SP1

Minimum requirements

The minimum requirements for a Windows system and user Hybrid Runbook Worker are:

- Windows PowerShell 5.1 ([download WMF 5.1](#)). PowerShell Core is not supported.
- .NET Framework 4.6.2 or later
- Two cores
- 4 GB of RAM

- Port 443 (outbound)

Network configuration

For networking requirements for the Hybrid Runbook Worker, see [Configuring your network](#).

Adding a machine to a Hybrid Runbook Worker group

You can add the worker machine to a Hybrid Runbook Worker group in one of your Automation accounts. For machines hosting the system Hybrid Runbook worker managed by Update Management, they can be added to a Hybrid Runbook Worker group. But you must use the same Automation account for both Update Management and the Hybrid Runbook Worker group membership.

NOTE

Azure Automation [Update Management](#) automatically installs the system Hybrid Runbook Worker on an Azure or non-Azure machine that's enabled for Update Management. However, this worker is not registered with any Hybrid Runbook Worker groups in your Automation account. To run your runbooks on those machines, you need to add them to a Hybrid Runbook Worker group. Follow step 6 under the section [Manual deployment](#) to add it to a group.

Enable for management with Azure Automation State Configuration

For information about enabling machines for management with Azure Automation State Configuration, see [Enable machines for management by Azure Automation State Configuration](#).

NOTE

To manage the configuration of machines that support the Hybrid Runbook Worker role with Desired State Configuration (DSC), you must add the machines as DSC nodes.

Installation options

To install and configure a Windows user Hybrid Runbook Worker, you can use one of the following methods.

- Use a provided PowerShell script to completely [automate](#) the process of configuring one or more Windows machines. This is the recommended method for machines in your datacenter or another cloud environment.
- Manually import the Automation solution, install the Log Analytics agent for Windows, and configure the worker role on the machine.

Automated deployment

There are two methods to automatically deploy a Hybrid Runbook Worker. You can import a runbook from the Runbook Gallery in the Azure portal and run it, or you can manually download a script from the PowerShell Gallery.

Importing a runbook from the Runbook Gallery

The import procedure is described in detail in [Import runbooks from GitHub with the Azure portal](#). The name of the runbook to import is **Create Automation Windows HybridWorker**.

The runbook uses the following parameters.

PARAMETER	STATUS	DESCRIPTION
Location	Mandatory	The Location of the automation account in which the script is executed.

PARAMETER	STATUS	DESCRIPTION
<code>ResourceGroupName</code>	Mandatory	The resource group for your Automation account.
<code>AccountName</code>	Mandatory	The Automation account name in which the Hybrid Run Worker will be registered.
<code>CreateLA</code>	Mandatory	If true, uses the value of <code>WorkspaceName</code> to create a Log Analytics workspace. If false, the value of <code>WorkspaceName</code> must refer to an existing workspace.
<code>LAllocation</code>	Optional	The location where the Log Analytics workspace will be created, or where it already exists.
<code>WorkspaceName</code>	Optional	The name of the Log Analytics workspace to use.
<code>CreateVM</code>	Mandatory	If true, use the value of <code>VMName</code> as the name of a new VM. If false, use <code>VMName</code> to find and register existing VM.
<code>VMName</code>	Optional	The name of the virtual machine that's either created or registered, depending on the value of <code>CreateVM</code> .
<code>VMIImage</code>	Optional	The name of the VM image to be created.
<code>VMlocation</code>	Optional	Location of the VM that's either created or registered. If this location is not specified, the value of <code>LAllocation</code> is used.
<code>RegisterHW</code>	Mandatory	If true, register the VM as a hybrid worker.
<code>WorkerGroupName</code>	Mandatory	Name of the Hybrid Worker Group.

Download a script from the PowerShell Gallery

This automated deployment method uses the PowerShell script `New-OnPremiseHybridWorker.ps1` to automate and configure the Windows Hybrid Runbook Worker role. It performs the following:

- Installs the necessary modules
- Signs in with your Azure account
- Verifies the existence of specified resource group and Automation account
- Creates references to Automation account attributes
- Creates an Azure Monitor Log Analytics workspace if not specified
- Enable the Azure Automation solution in the workspace
- Download and install the Log Analytics agent for Windows

- Register the machine as Hybrid Runbook Worker

Perform the following steps to install the role on your Windows machine using the script.

1. Download the **New-OnPremiseHybridWorker.ps1** script from the [PowerShell Gallery](#). After you have downloaded the script, copy or run it on the target machine. The script uses the following parameters.

PARAMETER	STATUS	DESCRIPTION
<code>AAResourceGroupName</code>	Mandatory	The name of the resource group that's associated with your Automation account.
<code>AutomationAccountName</code>	Mandatory	The name of your Automation account.
<code>Credential</code>	Optional	The credentials to use when logging in to the Azure environment.
<code>HybridGroupName</code>	Mandatory	The name of a Hybrid Runbook Worker group that you specify as a target for the runbooks that support this scenario.
<code>OMSResourceGroupName</code>	Optional	The name of the resource group for the Log Analytics workspace. If this resource group is not specified, the value of <code>AAResourceGroupName</code> is used.
<code>SubscriptionID</code>	Mandatory	The identifier of the Azure subscription associated with your Automation account.
<code>TenantID</code>	Optional	The identifier of the tenant organization associated with your Automation account.
<code>WorkspaceName</code>	Optional	The Log Analytics workspace name. If you don't have a Log Analytics workspace, the script creates and configures one.

2. Open an elevated 64-bit PowerShell command prompt.
3. From the PowerShell command prompt, browse to the folder that contains the script that you downloaded. Change the values for the parameters `AutomationAccountName`, `AAResourceGroupName`, `OMSResourceGroupName`, `HybridGroupName`, `SubscriptionID`, and `WorkspaceName`. Then run the script.

You're prompted to authenticate with Azure after you run the script. You must sign in with an account that's a member of the **Subscription Admins** role and co-administrator of the subscription.

```
$NewOnPremiseHybridWorkerParameters = @{
    AutomationAccountName = <nameOfAutomationAccount>
    AAResourceGroupName = <nameOfResourceGroup>
    OMSResourceGroupName = <nameOfResourceGroup>
    HybridGroupName = <nameOfHRWGroup>
    SubscriptionID = <subscriptionId>
    WorkspaceName = <nameOfLogAnalyticsWorkspace>
}
.\New-OnPremiseHybridWorker.ps1 @NewOnPremiseHybridWorkerParameters
```

4. You're prompted to agree to install NuGet, and to authenticate with your Azure credentials. If you don't have the latest NuGet version, you can download it from [Available NuGet Distribution Versions](#).
5. Verify the deployment after the script is finished. From the **Hybrid Runbook Worker Groups** page in your Automation account, under the **User hybrid runbook workers group** tab, it shows the new group and the number of members. If it's an existing group, the number of members is incremented. You can select the group from the list on the page, from the left-hand menu choose **Hybrid Workers**. On the **Hybrid Workers** page, you can see each member of the group listed.

Manual deployment

To install and configure a Windows Hybrid Runbook Worker, perform the following steps.

1. Enable the Azure Automation solution in your Log Analytics workspace by running the following command in an elevated PowerShell command prompt or in Cloud Shell in the [Azure portal](#).

```
Set-AzOperationalInsightsIntelligencePack -ResourceGroupName <resourceGroupName> -WorkspaceName <workspaceName> -IntelligencePackName "AzureAutomation" -Enabled $true
```

2. Deploy the Log Analytics agent to the target machine.

- For Azure VMs, install the Log Analytics agent for Windows using the [virtual machine extension for Windows](#). The extension installs the Log Analytics agent on Azure virtual machines, and enrolls virtual machines into an existing Log Analytics workspace. You can use an Azure Resource Manager template, PowerShell, or Azure Policy to assign the [Deploy Log Analytics agent for Linux or Windows VMs](#) built-in policy definition. Once the agent is installed, the machine can be added to a Hybrid Runbook Worker group in your Automation account.
- For non-Azure machines, you can install the Log Analytics agent using [Azure Arc-enabled servers](#). Arc-enabled servers support deploying the Log Analytics agent using the following methods:

- Using the VM extensions framework.

This feature in Azure Arc-enabled servers allows you to deploy the Log Analytics agent VM extension to a non-Azure Windows and/or Linux server. VM extensions can be managed using the following methods on your hybrid machines or servers managed by Arc-enabled servers:

- The [Azure portal](#)
- The [Azure CLI](#)
- [Azure PowerShell](#)
- [Azure Resource Manager templates](#)
- Using Azure Policy.

Using this approach, you use the Azure Policy [Deploy Log Analytics agent to Linux or Windows Azure Arc machines](#) built-in policy definition to audit if the Arc-enabled server has

the Log Analytics agent installed. If the agent is not installed, it automatically deploys it using a remediation task. Alternatively, if you plan to monitor the machines with Azure Monitor for VMs, instead use the [Enable Azure Monitor for VMs](#) initiative to install and configure the Log Analytics agent.

We recommend installing the Log Analytics agent for Windows or Linux using Azure Policy.

3. Verify agent is reporting to workspace

The Log Analytics agent for Windows connects machines to an Azure Monitor Log Analytics workspace. When you install the agent on your machine and connect it to your workspace, it automatically downloads the components that are required for the Hybrid Runbook Worker.

When the agent has successfully connected to your Log Analytics workspace after a few minutes, you can run the following query to verify that it is sending heartbeat data to the workspace.

```
Heartbeat  
| where Category == "Direct Agent"  
| where TimeGenerated > ago(30m)
```

In the search results, you should see heartbeat records for the machine, indicating that it is connected and reporting to the service. By default, every agent forwards a heartbeat record to its assigned workspace. Use the following steps to complete the agent installation and setup.

4. Confirm the version of the Hybrid Runbook Worker on the machine hosting the Log Analytics agent, browse to `C:\Program Files\Microsoft Monitoring Agent\Agent\AzureAutomation\` and note the **version** subfolder. This folder will appear on the machine several minutes after the solution is enabled in the workspace.
5. Install the runbook environment and connect to Azure Automation. When you configure an agent to report to a Log Analytics workspace and import the **Automation** solution, the solution pushes down the `HybridRegistration` PowerShell module. This module contains the `Add-HybridRunbookWorker` cmdlet. Use this cmdlet to install the runbook environment on the machine and register it with Azure Automation.

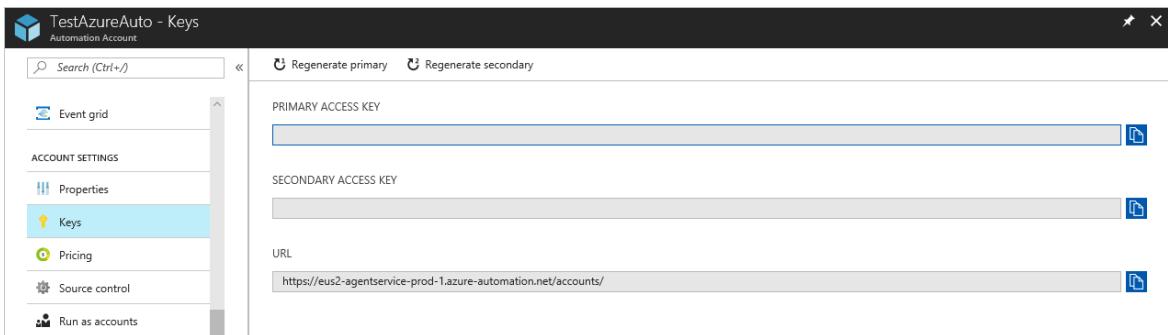
Open a PowerShell session in Administrator mode and run the following commands to import the module.

```
cd "C:\Program Files\Microsoft Monitoring Agent\Agent\AzureAutomation\<version>\HybridRegistration"  
Import-Module .\HybridRegistration.psd1
```

6. Run the `Add-HybridRunbookWorker` cmdlet specifying the values for the parameters `Url`, `Key`, and `GroupName`.

```
Add-HybridRunbookWorker -GroupName <String> -Url <Url> -Key <String>
```

You can get the information required for the parameters `Url` and `Key` from the **Keys** page in your Automation account. Select **Keys** under the **Account settings** section from the left-hand side of the page.



- For the `Url` parameter, copy the value for URL.
 - For the `key` parameter, copy the value for PRIMARY ACCESS KEY.
 - For the `GroupName` parameter, use the name of the Hybrid Runbook Worker group. If this group already exists in the Automation account, the current machine is added to it. If this group doesn't exist, it's added.
 - If required, set the `verbose` parameter to receive details about the installation.
7. Verify the deployment after the command is completed. From the **Hybrid Runbook Worker Groups** page in your Automation account, under the **User hybrid runbook workers group** tab, it shows the new or existing group and the number of members. If it's an existing group, the number of members is incremented. You can select the group from the list on the page, from the left-hand menu choose **Hybrid Workers**. On the **Hybrid Workers** page, you can see each member of the group listed.

Install PowerShell modules

Runbooks can use any of the activities and cmdlets defined in the modules installed in your Azure Automation environment. As these modules are not automatically deployed to on-premises machines, you must install them manually. The exception is the Azure module. This module is installed by default and provides access to cmdlets for all Azure services and activities for Azure Automation.

Because the primary purpose of the Hybrid Runbook Worker is to manage local resources, you most likely need to install the modules that support these resources, particularly the `PowerShellGet` module. For information on installing Windows PowerShell modules, see [Windows PowerShell](#).

Modules that are installed must be in a location referenced by the `PSModulePath` environment variable so that the hybrid worker can automatically import them. For more information, see [Install Modules in PSModulePath](#).

Remove the Hybrid Runbook Worker

1. In the Azure portal, go to your Automation account.
2. Under Account Settings, select Keys and note the values for URL and Primary Access Key.
3. Open a PowerShell session in Administrator mode and run the following command with your URL and primary access key values. Use the `Verbose` parameter for a detailed log of the removal process. To remove stale machines from your Hybrid Worker group, use the optional `MachineName` parameter.

```
Remove-HybridRunbookWorker -Url <URL> -Key <primaryAccessKey> -MachineName <computerName>
```

Remove a Hybrid Worker group

To remove a Hybrid Runbook Worker group, you first need to remove the Hybrid Runbook Worker from every machine that is a member of the group. Then use the following steps to remove the group:

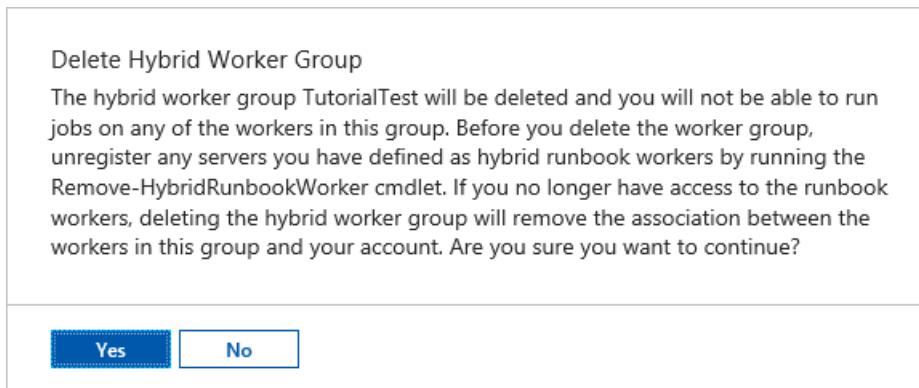
1. Open the Automation account in the Azure portal.
2. Select **Hybrid worker groups** under **Process Automation**. Select the group that you want to delete.
The properties page for that group appears.

The screenshot shows the Azure portal interface for managing hybrid worker groups. On the left, there's a sidebar with a search bar and links for Overview, Properties, Hybrid worker group settings, and Hybrid Workers. The main content area is titled 'Delete' and shows the 'Essentials' section for a group named 'TutorialTest'. The details include:

Resource group	Account
TestAzureAuto	TestAzureAuto
Location	Subscription name
eastus2	Microsoft Azure
Last registration time	Last seen time
12/11/2017 12:56 PM	12/11/2017 1:27 PM
Group type	Run As
User	--

Below this is a 'Details' section for 'Hybrid Workers', which shows a count of 1 and a small icon.

3. On the properties page for the selected group, select **Delete**. A message asks you to confirm this action. Select **Yes** if you're sure that you want to continue.



This process can take several seconds to finish. You can track its progress under **Notifications** from the menu.

Next steps

- To learn how to configure your runbooks to automate processes in your on-premises datacenter or other cloud environment, see [Run runbooks on a Hybrid Runbook Worker](#).
- To learn how to troubleshoot your Hybrid Runbook Workers, see [Troubleshoot Hybrid Runbook Worker issues](#).

Deploy a Linux Hybrid Runbook Worker

8/24/2021 • 10 minutes to read • [Edit Online](#)

You can use the user Hybrid Runbook Worker feature of Azure Automation to run runbooks directly on the Azure or non-Azure machine, including servers registered with [Azure Arc-enabled servers](#). From the machine or server that's hosting the role, you can run runbooks directly on it and against resources in the environment to manage those local resources.

The Linux Hybrid Runbook Worker executes runbooks as a special user that can be elevated for running commands that need elevation. Azure Automation stores and manages runbooks and then delivers them to one or more designated machines. This article describes how to install the Hybrid Runbook Worker on a Linux machine, how to remove the worker, and how to remove a Hybrid Runbook Worker group.

After you successfully deploy a runbook worker, review [Run runbooks on a Hybrid Runbook Worker](#) to learn how to configure your runbooks to automate processes in your on-premises datacenter or other cloud environment.

Prerequisites

Before you start, make sure that you have the following.

A Log Analytics workspace

The Hybrid Runbook Worker role depends on an Azure Monitor Log Analytics workspace to install and configure the role. You can create it through [Azure Resource Manager](#), through [PowerShell](#), or in the [Azure portal](#).

If you don't have an Azure Monitor Log Analytics workspace, review the [Azure Monitor Log design guidance](#) before you create the workspace.

Log Analytics agent

The Hybrid Runbook Worker role requires the [Log Analytics agent](#) for the supported Linux operating system. For servers or machines hosted outside of Azure, you can install the Log Analytics agent using [Azure Arc-enabled servers](#).

NOTE

After installing the Log Analytics agent for Linux, you should not change the permissions of the `sudoers.d` folder or its ownership. Sudo permission is required for the `nxautomation` account, which is the user context the Hybrid Runbook Worker runs under. The permissions should not be removed. Restricting this to certain folders or commands may result in a breaking change.

Supported Linux operating systems

The Hybrid Runbook Worker feature supports the following distributions. All operating systems are assumed to be x64. x86 is not supported for any operating system.

- Amazon Linux 2012.09 to 2015.09
- CentOS Linux 5, 6, 7, and 8
- Oracle Linux 6, 7, and 8
- Red Hat Enterprise Linux Server 5, 6, 7, and 8
- Debian GNU/Linux 6, 7, and 8

- Ubuntu 12.04 LTS, 14.04 LTS, 16.04 LTS, 18.04, and 20.04 LTS
- SUSE Linux Enterprise Server 12, 15, and 15.1 (SUSE did not release versions numbered 13 or 14)

IMPORTANT

Before enabling the Update Management feature, which depends on the system Hybrid Runbook Worker role, confirm the distributions it supports [here](#).

Minimum requirements

The minimum requirements for a Linux system and user Hybrid Runbook Worker are:

- Two cores
- 4 GB of RAM
- Port 443 (outbound)

REQUIRED PACKAGE	DESCRIPTION	MINIMUM VERSION
Glibc	GNU C Library	2.5-12
Openssl	OpenSSL Libraries	1.0 (TLS 1.1 and TLS 1.2 are supported)
Curl	cURL web client	7.15.5
Python-ctypes	Foreign function library for Python	Python 2.x or Python 3.x are required
PAM	Pluggable Authentication Modules	

OPTIONAL PACKAGE	DESCRIPTION	MINIMUM VERSION
PowerShell Core	To run PowerShell runbooks, PowerShell Core needs to be installed. See Installing PowerShell Core on Linux to learn how to install it.	6.0.0

Adding a machine to a Hybrid Runbook Worker group

You can add the worker machine to a Hybrid Runbook Worker group in one of your Automation accounts. For machines hosting the system Hybrid Runbook worker managed by Update Management, they can be added to a Hybrid Runbook Worker group. But you must use the same Automation account for both Update Management and the Hybrid Runbook Worker group membership.

NOTE

Azure Automation [Update Management](#) automatically installs the system Hybrid Runbook Worker on an Azure or non-Azure machine that's enabled for Update Management. However, this worker is not registered with any Hybrid Runbook Worker groups in your Automation account. To run your runbooks on those machines, you need to add them to a Hybrid Runbook Worker group. Follow step 4 under the section [Install a Linux Hybrid Runbook Worker](#) to add it to a group.

Supported Linux hardening

The following are not yet supported:

- CIS

Supported runbook types

Linux Hybrid Runbook Workers support a limited set of runbook types in Azure Automation, and they are described in the following table.

RUNBOOK TYPE	SUPPORTED
Python 3 (preview)	Yes, required for these distros only: SUSE LES 15, RHEL 8 and CentOS 8
Python 2	Yes, for any distro that doesn't require Python 3 ¹
PowerShell	Yes ²
PowerShell Workflow	No
Graphical	No
Graphical PowerShell Workflow	No

¹See [Supported Linux operating systems](#).

²PowerShell runbooks require PowerShell Core to be installed on the Linux machine. See [Installing PowerShell Core on Linux](#) to learn how to install it.

Network configuration

For networking requirements for the Hybrid Runbook Worker, see [Configuring your network](#).

Install a Linux Hybrid Runbook Worker

There are two methods to deploy a Hybrid Runbook Worker. You can import and run a runbook from the Runbook Gallery in the Azure portal, or you can manually run a series of PowerShell commands to accomplish the same task.

Importing a runbook from the Runbook Gallery

The import procedure is described in detail in [Import runbooks from GitHub with the Azure portal](#). The name of the runbook to import is **Create Automation Linux HybridWorker**.

The runbook uses the following parameters.

PARAMETER	STATUS	DESCRIPTION
Location	Mandatory	The location for the Log Analytics workspace.
ResourceGroupName	Mandatory	The resource group for your Automation account.
AccountName	Mandatory	The Automation account name in which the Hybrid Run Worker will be registered.

PARAMETER	STATUS	DESCRIPTION
CreateLA	Mandatory	If true, uses the value of <code>WorkspaceName</code> to create a Log Analytics workspace. If false, the value of <code>WorkspaceName</code> must refer to an existing workspace.
LAllocation	Optional	The location where the Log Analytics workspace will be created, or where it already exists.
WorkspaceName	Optional	The name of the Log Analytics workspace to be created or used.
CreateVM	Mandatory	If true, use the value of <code>VMName</code> as the name of a new VM. If false, use <code>VMName</code> to find and register existing VM.
VMName	Optional	The name of the virtual machine that's either created or registered, depending on the value of <code>CreateVM</code> .
VMIImage	Optional	The name of the VM image to be created.
VMlocation	Optional	Location of the VM that's either created or registered. If this location is not specified, the value of <code>LAllocation</code> is used.
RegisterHW	Mandatory	If true, register the VM as a hybrid worker.
WorkerGroupName	Mandatory	Name of the Hybrid Worker Group.

Manually run PowerShell commands

To install and configure a Linux Hybrid Runbook Worker, perform the following steps.

1. Enable the Azure Automation solution in your Log Analytics workspace by running the following command in an elevated PowerShell command prompt or in Cloud Shell in the [Azure portal](#):

```
Set-AzOperationalInsightsIntelligencePack -ResourceGroupName <resourceGroupName> -WorkspaceName <workspaceName> -IntelligencePackName "AzureAutomation" -Enabled $true
```

2. Deploy the Log Analytics agent to the target machine.

- For Azure VMs, install the Log Analytics agent for Linux using the [virtual machine extension for Linux](#). The extension installs the Log Analytics agent on Azure virtual machines, and enrolls virtual machines into an existing Log Analytics workspace. You can use an Azure Resource Manager template, the Azure CLI, or Azure Policy to assign the [Deploy Log Analytics agent for Linux or Windows VMs](#) built-in policy definition. Once the agent is installed, the machine can be added to a Hybrid Runbook Worker group in your Automation account.
- For non-Azure machines, you can install the Log Analytics agent using [Azure Arc-enabled servers](#).

Arc-enabled servers support deploying the Log Analytics agent using the following methods:

- Using the VM extensions framework.

This feature in Azure Arc-enabled servers allows you to deploy the Log Analytics agent VM extension to a non-Azure Windows and/or Linux server. VM extensions can be managed using the following methods on your hybrid machines or servers managed by Arc-enabled servers:

- The [Azure portal](#)
- The [Azure CLI](#)
- [Azure PowerShell](#)
- Azure [Resource Manager templates](#)
- Using Azure Policy.

Using this approach, you use the Azure Policy [Deploy Log Analytics agent to Linux or Windows Azure Arc machines](#) built-in policy definition to audit if the Arc-enabled server has the Log Analytics agent installed. If the agent is not installed, it automatically deploys it using a remediation task. Alternatively, if you plan to monitor the machines with Azure Monitor for VMs, instead use the [Enable Azure Monitor for VMs](#) initiative to install and configure the Log Analytics agent.

We recommend installing the Log Analytics agent for Windows or Linux using Azure Policy.

NOTE

To manage the configuration of machines that support the Hybrid Runbook Worker role with Desired State Configuration (DSC), you must add the machines as DSC nodes.

NOTE

The [nxautomation account](#) with the corresponding sudo permissions must be present during installation of the Linux Hybrid Worker. If you try to install the worker and the account is not present or doesn't have the appropriate permissions, the installation fails.

3. Verify agent is reporting to workspace.

The Log Analytics agent for Linux connects machines to an Azure Monitor Log Analytics workspace. When you install the agent on your machine and connect it to your workspace, it automatically downloads the components that are required for the Hybrid Runbook Worker.

When the agent has successfully connected to your Log Analytics workspace after a few minutes, you can run the following query to verify that it is sending heartbeat data to the workspace.

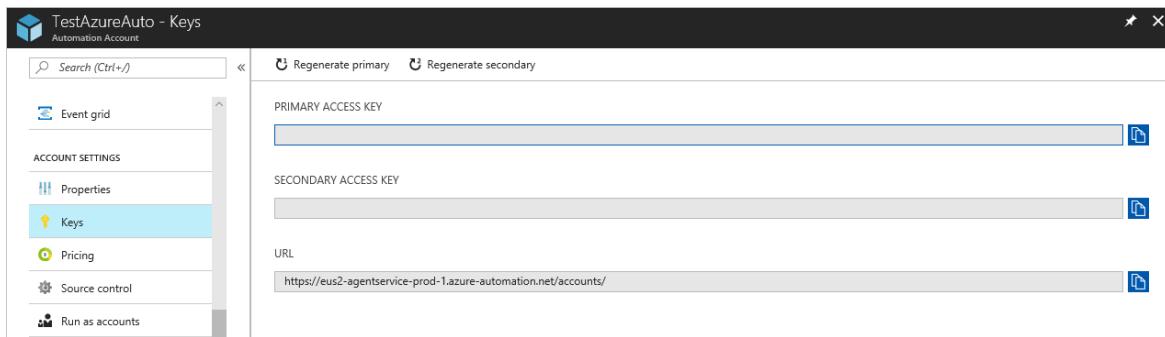
```
Heartbeat  
| where Category == "Direct Agent"  
| where TimeGenerated > ago(30m)
```

In the search results, you should see heartbeat records for the machine, indicating that it is connected and reporting to the service. By default, every agent forwards a heartbeat record to its assigned workspace.

4. Run the following command to add the machine to a Hybrid Runbook Worker group, specifying the values for the parameters `-w`, `-k`, `-g`, and `-e`.

You can get the information required for parameters `-k` and `-e` from the [Keys](#) page in your

Automation account. Select **Keys** under the **Account settings** section from the left-hand side of the page.



- For the `-e` parameter, copy the value for URL.
- For the `-k` parameter, copy the value for PRIMARY ACCESS KEY.
- For the `-g` parameter, specify the name of the Hybrid Runbook Worker group that the new Linux Hybrid Runbook worker should join. If this group already exists in the Automation account, the current machine is added to it. If this group doesn't exist, it is created with that name.
- For the `-w` parameter, specify your Log Analytics workspace ID.

```
sudo python
/opt/microsoft/omsconfig/modules/nxOMSAutomationWorker/DSCResources/MSFT_nxOMSAutomationWorkerResource/automationworker/scripts/onboarding.py --register -w <logAnalyticsworkspaceId> -k
<automationSharedKey> -g <hybridGroupName> -e <automationEndpoint>
```

5. Verify the deployment after the script is completed. From the **Hybrid Runbook Worker Groups** page in your Automation account, under the **User hybrid runbook workers group** tab, it shows the new or existing group and the number of members. If it's an existing group, the number of members is incremented. You can select the group from the list on the page, from the left-hand menu choose **Hybrid Workers**. On the **Hybrid Workers** page, you can see each member of the group listed.

NOTE

If you are using the Log Analytics virtual machine extension for Linux for an Azure VM, we recommend setting `autoUpgradeMinorVersion` to `false` as auto-upgrading versions can cause issues with the Hybrid Runbook Worker. To learn how to upgrade the extension manually, see [Azure CLI deployment](#).

Turn off signature validation

By default, Linux Hybrid Runbook Workers require signature validation. If you run an unsigned runbook against a worker, you see a `Signature validation failed` error. To turn off signature validation, run the following command. Replace the second parameter with your Log Analytics workspace ID.

```
sudo python
/opt/microsoft/omsconfig/modules/nxOMSAutomationWorker/DSCResources/MSFT_nxOMSAutomationWorkerResource/automationworker/scripts/require_runbook_signature.py --false <logAnalyticsworkspaceId>
```

Remove the Hybrid Runbook Worker

You can use the command `ls /var/opt/microsoft/omsagent` on the Hybrid Runbook Worker to get the workspace ID. A folder is created that is named with the workspace ID.

```
sudo python onboarding.py --deregister --endpoint=<URL> --key=<PrimaryAccessKey> --groupname="Example" -  
-workspaceid=<workspaceId>"
```

NOTE

This script doesn't remove the Log Analytics agent for Linux from the machine. It only removes the functionality and configuration of the Hybrid Runbook Worker role.

Remove a Hybrid Worker group

To remove a Hybrid Runbook Worker group of Linux machines, you use the same steps as for a Windows hybrid worker group. See [Remove a Hybrid Worker group](#).

Next steps

- To learn how to configure your runbooks to automate processes in your on-premises datacenter or other cloud environment, see [Run runbooks on a Hybrid Runbook Worker](#).
- To learn how to troubleshoot your Hybrid Runbook Workers, see [Troubleshoot Hybrid Runbook Worker issues - Linux](#).

Run runbooks on a Hybrid Runbook Worker

8/26/2021 • 12 minutes to read • [Edit Online](#)

Runbooks that run on a [Hybrid Runbook Worker](#) typically manage resources on the local computer or against resources in the local environment where the worker is deployed. Runbooks in Azure Automation typically manage resources in the Azure cloud. Even though they are used differently, runbooks that run in Azure Automation and runbooks that run on a Hybrid Runbook Worker are identical in structure.

When you author a runbook to run on a Hybrid Runbook Worker, you should edit and test the runbook on the machine that hosts the worker. The host machine has all the PowerShell modules and network access required to manage the local resources. Once you test the runbook on the Hybrid Runbook Worker machine, you can then upload it to the Azure Automation environment, where it can be run on the worker.

Plan for Azure services protected by firewall

Enabling the Azure Firewall on [Azure Storage](#), [Azure Key Vault](#), or [Azure SQL](#) blocks access from Azure Automation runbooks for those services. Access will be blocked even when the firewall exception to allow trusted Microsoft services is enabled, as Automation is not a part of the trusted services list. With an enabled firewall, access can only be made by using a Hybrid Runbook Worker and a [virtual network service endpoint](#).

Plan runbook job behavior

Azure Automation handles jobs on Hybrid Runbook Workers differently from jobs run in Azure sandboxes. If you have a long-running runbook, make sure that it's resilient to possible restart. For details of the job behavior, see [Hybrid Runbook Worker jobs](#).

Jobs for Hybrid Runbook Workers run under the local **System** account on Windows, or the **nxautomation** account on Linux. For Linux, verify the **nxautomation** account has access to the location where the runbook modules are stored. To ensure **nxautomation** account access:

- When you use the `Install-Module` cmdlet, be sure to specify `AllUsers` for the `Scope` parameter.
- When you use `pip install`, `apt install` or other method for installing packages on Linux, ensure the package is installed for all users. For example `sudo -H pip install <package_name>`.

For more information on PowerShell on Linux, see [Known Issues for PowerShell on Non-Windows Platforms](#).

Configure runbook permissions

Define permissions for your runbook to run on the Hybrid Runbook Worker in the following ways:

- Have the runbook provide its own authentication to local resources.
- Configure authentication using [managed identities for Azure resources](#).
- Specify a Run As account to provide a user context for all runbooks.

Use runbook authentication to local resources

If preparing a runbook that provides its own authentication to resources, use [credential](#) and [certificate](#) assets in your runbook. There are several cmdlets that allow you to specify credentials so that the runbook can authenticate to different resources. The following example shows a portion of a runbook that restarts a computer. It retrieves credentials from a credential asset and the name of the computer from a variable asset and then uses these values with the `Restart-Computer` cmdlet.

```
$Cred = Get-AutomationPSCredential -Name "MyCredential"
$Computer = Get-AutomationVariable -Name "ComputerName"

Restart-Computer -ComputerName $Computer -Credential $Cred
```

You can also use an [InlineScript](#) activity. [InlineScript](#) allows you to run blocks of code on another computer with credentials.

Use runbook authentication with managed identities

Hybrid Runbook Workers on Azure virtual machines can use managed identities to authenticate to Azure resources. Using managed identities for Azure resources instead of Run As accounts provides benefits because you don't need to:

- Export the Run As certificate and then import it into the Hybrid Runbook Worker.
- Renew the certificate used by the Run As account.
- Handle the Run As connection object in your runbook code.

Follow the next steps to use a managed identity for Azure resources on a Hybrid Runbook Worker:

1. Create an Azure VM.
2. Configure managed identities for Azure resources on the VM. See [Configure managed identities for Azure resources on a VM using the Azure portal](#).
3. Give the VM access to a resource group in Resource Manager. Refer to [Use a Windows VM system-assigned managed identity to access Resource Manager](#).
4. Install the Hybrid Runbook Worker on the VM. See [Deploy a Windows Hybrid Runbook Worker](#) or [Deploy a Linux Hybrid Runbook Worker](#).
5. Update the runbook to use the [Connect-AzAccount](#) cmdlet with the [Identity](#) parameter to authenticate to Azure resources. This configuration reduces the need to use a Run As account and perform the associated account management.

```
# Connect to Azure using the managed identities for Azure resources identity configured on the Azure
# VM that is hosting the hybrid runbook worker
Connect-AzAccount -Identity

# Get all VM names from the subscription
Get-AzVM | Select Name
```

NOTE

[Connect-AzAccount -Identity](#) works for a Hybrid Runbook Worker using a system-assigned identity and a single user-assigned identity. If you use multiple user-assigned identities on the Hybrid Runbook Worker, your runbook must specify the [AccountId](#) parameter for [Connect-AzAccount](#) to select a specific user-assigned identity.

Use runbook authentication with Run As account

Instead of having your runbook provide its own authentication to local resources, you can specify a Run As account for a Hybrid Runbook Worker group. To specify a Run As account, you must define a [credential asset](#) that has access to local resources. These resources include certificate stores and all runbooks run under these credentials on a Hybrid Runbook Worker in the group.

- The user name for the credential must be in one of the following formats:

- domain\username
- username@domain
- username (for accounts local to the on-premises computer)
- To use the PowerShell runbook **Export-RunAsCertificateToHybridWorker**, you need to install the Az modules for Azure Automation on the local machine.

Use a credential asset to specify a Run As account

Use the following procedure to specify a Run As account for a Hybrid Runbook Worker group:

1. Create a [credential asset](#) with access to local resources.
2. Open the Automation account in the Azure portal.
3. Select **Hybrid Worker Groups**, and then select the specific group.
4. Select **All settings**, followed by **Hybrid worker group settings**.
5. Change the value of **Run As** from **Default** to **Custom**.
6. Select the credential and click **Save**.

Install Run As account certificate

As part of your automated build process for deploying resources in Azure, you might require access to on-premises systems to support a task or set of steps in your deployment sequence. To provide authentication against Azure using the Run As account, you must install the Run As account certificate.

NOTE

This PowerShell runbook currently does not run on Linux machines. It runs only on Windows machines.

The following PowerShell runbook, called **Export-RunAsCertificateToHybridWorker**, exports the Run As certificate from your Azure Automation account. The runbook downloads and imports the certificate into the local machine certificate store on a Hybrid Runbook Worker that is connected to the same account. Once it completes that step, the runbook verifies that the worker can successfully authenticate to Azure using the Run As account.

NOTE

This PowerShell runbook is not designed or intended to be run outside of your Automation account as a script on the target machine.

```
<#PSScriptInfo
.VERSION 1.0
.GUID 3a796b9a-623d-499d-86c8-c249f10a6986
.AUTHOR Azure Automation Team
.COMPANYNAME Microsoft
.COPYRIGHT
.TAGS Azure Automation
.LICENSEURI
.PROJECTURI
.ICONURI
.EXTERNALMODULEDEPENDENCIES
.REQUIREDSCRIPTS
.EXTERNALSCRIPTDEPENDENCIES
.RELEASENOTES
#>

<#
.SYNOPSIS
Exports the Run As certificate from an Azure Automation account to a hybrid worker in that account.
```

.DESCRIPTION

This runbook exports the Run As certificate from an Azure Automation account to a hybrid worker in that account. Run this runbook on the hybrid worker where you want the certificate installed. This allows the use of the AzureRunAsConnection to authenticate to Azure and manage Azure resources from runbooks running on the hybrid worker.

.EXAMPLE

```
.\Export-RunAsCertificateToHybridWorker
```

.NOTES

LASTEDIT: 2016.10.13

#>

```
# Generate the password used for this certificate
Add-Type -AssemblyName System.Web -ErrorAction SilentlyContinue | Out-Null
$Password = [System.Web.Security.Membership]::GeneratePassword(25, 10)

# Stop on errors
$ErrorActionPreference = 'stop'

# Get the management certificate that will be used to make calls into Azure Service Management resources
$RunAsCert = Get-AutomationCertificate -Name "AzureRunAsCertificate"

# location to store temporary certificate in the Automation service host
$CertPath = Join-Path $env:temp "AzureRunAsCertificate.pfx"

# Save the certificate
$Cert = $RunAsCert.Export("pfx",$Password)
Set-Content -Value $Cert -Path $CertPath -Force -Encoding Byte | Write-Verbose

Write-Output ("Importing certificate into $env:computername local machine root store from " + $CertPath)
$SecurePassword = ConvertTo-SecureString $Password -AsPlainText -Force
Import-PfxCertificate -FilePath $CertPath -CertStoreLocation Cert:\LocalMachine\My -Password $SecurePassword
-Exportable | Write-Verbose

# Test to see if authentication to Azure Resource Manager is working
$RunAsConnection = Get-AutomationConnection -Name "AzureRunAsConnection"

Connect-AzAccount ` 
    -ServicePrincipal ` 
    -Tenant $RunAsConnection.TenantId ` 
    -ApplicationId $RunAsConnection.ApplicationId ` 
    -CertificateThumbprint $RunAsConnection.CertificateThumbprint | Write-Verbose

Set-AzContext -Subscription $RunAsConnection.SubscriptionID | Write-Verbose

# List automation accounts to confirm that Azure Resource Manager calls are working
Get-AzAutomationAccount | Select-Object AutomationAccountName
```

NOTE

For PowerShell runbooks, `Add-AzAccount` and `Add-AzureRMAccount` are aliases for `Connect-AzAccount`. When searching your library items, if you do not see `Connect-AzAccount`, you can use `Add-AzAccount`, or you can update your modules in your Automation account.

To finish preparing the Run As account:

1. Save the `Export-RunAsCertificateToHybridWorker` runbook to your computer with a `.ps1` extension.
2. Import it into your Automation account.
3. Edit the runbook, changing the value of the `Password` variable to your own password.
4. Publish the runbook.
5. Run the runbook, targeting the Hybrid Runbook Worker group that runs and authenticates runbooks using

the Run As account.

6. Examine the job stream to see that it reports the attempt to import the certificate into the local machine store, followed by multiple lines. This behavior depends on how many Automation accounts you define in your subscription and the degree of success of the authentication.

Work with signed runbooks on a Windows Hybrid Runbook Worker

You can configure a Windows Hybrid Runbook Worker to run only signed runbooks.

IMPORTANT

Once you've configured a Hybrid Runbook Worker to run only signed runbooks, unsigned runbooks fail to execute on the worker.

Create signing certificate

The following example creates a self-signed certificate that can be used for signing runbooks. This code creates the certificate and exports it so that the Hybrid Runbook Worker can import it later. The thumbprint is also returned for later use in referencing the certificate.

```
# Create a self-signed certificate that can be used for code signing
$SigningCert = New-SelfSignedCertificate -CertStoreLocation cert:\LocalMachine\my ` 
    -Subject "CN=contoso.com" ` 
    -KeyAlgorithm RSA ` 
    -KeyLength 2048 ` 
    -Provider "Microsoft Enhanced RSA and AES Cryptographic Provider" ` 
    -KeyExportPolicy Exportable ` 
    -KeyUsage DigitalSignature ` 
    -Type CodeSigningCert

# Export the certificate so that it can be imported to the hybrid workers
Export-Certificate -Cert $SigningCert -FilePath .\hybridworkersigningcertificate.cer

# Import the certificate into the trusted root store so the certificate chain can be validated
Import-Certificate -FilePath .\hybridworkersigningcertificate.cer -CertStoreLocation Cert:\LocalMachine\Root

# Retrieve the thumbprint for later use
$SigningCert.Thumbprint
```

Import certificate and configure workers for signature validation

Copy the certificate that you've created to each Hybrid Runbook Worker in a group. Run the following script to import the certificate and configure the workers to use signature validation on runbooks.

```
# Install the certificate into a location that will be used for validation.
New-Item -Path Cert:\LocalMachine\AutomationHybridStore
Import-Certificate -FilePath .\hybridworkersigningcertificate.cer -CertStoreLocation
Cert:\LocalMachine\AutomationHybridStore

# Import the certificate into the trusted root store so the certificate chain can be validated
Import-Certificate -FilePath .\hybridworkersigningcertificate.cer -CertStoreLocation Cert:\LocalMachine\Root

# Configure the hybrid worker to use signature validation on runbooks.
Set-HybridRunbookWorkerSignatureValidation -Enable $true -TrustedCertStoreLocation
"Cert:\LocalMachine\AutomationHybridStore"
```

Sign your runbooks using the certificate

With the Hybrid Runbook Workers configured to use only signed runbooks, you must sign runbooks that are to be used on the Hybrid Runbook Worker. Use the following sample PowerShell code to sign these runbooks.

```
$SigningCert = ( Get-ChildItem -Path cert:\LocalMachine\My\<CertificateThumbprint> )
Set-AuthenticodeSignature .\TestRunbook.ps1 -Certificate $SigningCert
```

When a runbook has been signed, you must import it into your Automation account and publish it with the signature block. To learn how to import runbooks, see [Import a runbook](#).

NOTE

Use only plaintext characters in your runbook code, including comments. Using characters with diacritical marks, like á or ñ, will result in an error. When Azure Automation downloads your code, the characters will be replaced by a question mark and the signing will fail with a "signature hash validation failure" message.

Work with signed runbooks on a Linux Hybrid Runbook Worker

To be able to work with signed runbooks, a Linux Hybrid Runbook Worker must have the [GPG](#) executable on the local machine.

IMPORTANT

Once you've configured a Hybrid Runbook Worker to run only signed runbooks, unsigned runbooks fail to execute on the worker.

You will perform the following steps to complete this configuration:

- Create a GPG keyring and keypair
- Make the keyring available to the Hybrid Runbook Worker
- Verify that signature validation is on
- Sign a runbook

Create a GPG keyring and keypair

To create the GPG keyring and keypair, use the Hybrid Runbook Worker [nxautomation account](#).

1. Use the sudo application to sign in as the **nxautomation** account.

```
sudo su - nxautomation
```

2. Once you are using **nxautomation**, generate the GPG keypair. GPG guides you through the steps. You must provide name, email address, expiration time, and passphrase. Then you wait until there is enough entropy on the machine for the key to be generated.

```
sudo gpg --generate-key
```

3. Because the GPG directory was generated with sudo, you must change its owner to **nxautomation** using the following command.

```
sudo chown -R nxautomation ~/.gnupg
```

Make the keyring available to the Hybrid Runbook Worker

Once the keyring has been created, make it available to the Hybrid Runbook Worker. Modify the settings file **home/nxautomation/state/worker.conf** to include the following example code under the file section

```
[worker-optional] .
```

```
gpg_public_keyring_path = /home/nxautomation/run/.gnupg/pubring.kbx
```

Verify that signature validation is on

If signature validation has been disabled on the machine, you must turn it on by running the following sudo command. Replace <LogAnalyticsworkspaceId> with your workspace ID.

```
sudo python  
/opt/microsoft/omsconfig/modules/nxOMSAutomationWorker/DSCResources/MSFT_nxOMSAutomationWorkerResource/automationworker/scripts/require_runbook_signature.py --true <LogAnalyticsworkspaceId>
```

Sign a runbook

Once you have configured signature validation, use the following GPG command to sign the runbook.

```
gpg --clear-sign <runbook name>
```

The signed runbook is called <runbook name>.asc.

You can now upload the signed runbook to Azure Automation and execute it like a regular runbook.

Start a runbook on a Hybrid Runbook Worker

[Start a runbook in Azure Automation](#) describes different methods for starting a runbook. Starting a runbook on a Hybrid Runbook Worker uses a **Run on** option that allows you to specify the name of a Hybrid Runbook Worker group. When a group is specified, one of the workers in that group retrieves and runs the runbook. If your runbook does not specify this option, Azure Automation runs the runbook as usual.

When you start a runbook in the Azure portal, you're presented with the **Run on** option for which you can select **Azure or Hybrid Worker**. If you select **Hybrid Worker**, you can choose the Hybrid Runbook Worker group from a dropdown.

When starting a runbook using PowerShell, use the **RunOn** parameter with the [Start-AzAutomationRunbook](#) cmdlet. The following example uses Windows PowerShell to start a runbook named **Test-Runbook** on a Hybrid Runbook Worker group named **MyHybridGroup**.

```
Start-AzAutomationRunbook -AutomationAccountName "MyAutomationAccount" -Name "Test-Runbook" -RunOn  
"MyHybridGroup"
```

Logging

To help troubleshoot issues with your runbooks running on a hybrid runbook worker, logs are stored locally in the following location:

- On Windows at `C:\ProgramData\Microsoft\System Center\Orchestrator\<version>\SMA\Sandboxes` for detailed job runtime process logging. High-level runbook job status events are written to the **Application and Services Logs\Microsoft-Automation\Operations** event log.
- On Linux, the user hybrid worker logs can be found at `/home/nxautomation/run/worker.log`, and system runbook worker logs can be found at `/var/opt/microsoft/omsagent/run/automationworker/worker.log`.

Next steps

- If your runbooks aren't completing successfully, review the troubleshooting guide for [runbook execution failures](#).
- For more information on PowerShell, including language reference and learning modules, see [PowerShell Docs](#).
- Learn about [using Azure Policy to manage runbook execution](#) with Hybrid Runbook Workers.
- For a PowerShell cmdlet reference, see [Az.Automation](#).

Use Azure Policy to enforce job execution on Hybrid Runbook Worker

5/25/2021 • 5 minutes to read • [Edit Online](#)

Starting a runbook on a Hybrid Runbook Worker uses a **Run on** option that allows you to specify the name of a Hybrid Runbook Worker group when initiating from the Azure portal, with the Azure PowerShell, or REST API. When a group is specified, one of the workers in that group retrieves and runs the runbook. If your runbook does not specify this option, Azure Automation runs the runbook in the Azure sandbox.

Anyone in your organization who is a member of the [Automation Job Operator](#) or higher can create runbook jobs. To manage runbook execution targeting a Hybrid Runbook Worker group in your Automation account, you can use [Azure Policy](#). This helps to enforce organizational standards and ensure your automation jobs are controlled and managed by those designated, and anyone cannot execute a runbook on an Azure sandbox, only on Hybrid Runbook workers.

A custom Azure Policy definition is included in this article to help you control these activities using the following Automation REST API operations. Specifically:

- [Create job](#)
- [Create job schedule](#)
- [Create webhook](#)

This policy is based on the `runOn` property. The policy validates the value of the property, which should contain the name of an existing Hybrid Runbook Worker group. If the value is null, it is interpreted as the create request for the job, job schedule, or webhook is intended for the Azure sandbox, and the request is denied.

Permissions required

You need to be a member of the [Owner](#) role at the subscription-level for permission to Azure Policy resources.

Create and assign the policy definition

Here we compose the policy rule and then assign it to either a management group or subscription, and optionally specify a resource group in the subscription. If you aren't yet familiar with the policy language, reference [policy definition structure](#) for how to structure the policy definition.

1. Use the following JSON snippet to create a JSON file with the name `AuditAutomationHRWJobExecution.json`.

```
{  
  "properties": {  
    "displayName": "Enforce job execution on Automation Hybrid Runbook Worker",  
    "description": "Enforce job execution on Hybrid Runbook Workers in your Automation account.",  
    "mode": "all",  
    "parameters": {  
      "effectType": {  
        "type": "string",  
        "defaultValue": "Deny",  
        "allowedValues": [  
          "Deny",  
          "Disabled"  
        ],  
        "metadata": {  
          "description": "The effect type for the policy rule."  
        }  
      }  
    }  
  }  
}
```

```

        "displayName": "Effect",
        "description": "Enable or disable execution of the policy"
    }
}
},
"policyRule": {
    "if": {
        "anyOf": [
            {
                "allOf": [
                    {
                        "field": "type",
                        "equals": "Microsoft.Automation/automationAccounts/jobs"
                    },
                    {
                        "value": "[length(field('Microsoft.Automation/automationAccounts/jobs/runOn'))]",
                        "less": 1
                    }
                ]
            },
            {
                "allOf": [
                    {
                        "field": "type",
                        "equals": "Microsoft.Automation/automationAccounts/webhooks"
                    },
                    {
                        "value": "[length(field('Microsoft.Automation/automationAccounts/webhooks/runOn'))]",
                        "less": 1
                    }
                ]
            },
            {
                "allOf": [
                    {
                        "field": "type",
                        "equals": "Microsoft.Automation/automationAccounts/jobSchedules"
                    },
                    {
                        "value": "
[length(field('Microsoft.Automation/automationAccounts/jobSchedules/runOn'))]",
                        "less": 1
                    }
                ]
            },
            {
                "then": {
                    "effect": "[parameters('effectType')]"
                }
            }
        ]
    }
}
}

```

- Run the following Azure PowerShell or Azure CLI command to create a policy definition using the AuditAutomationHRWJobExecution.json file.

- [Azure CLI](#)
- [PowerShell](#)

```

az policy definition create --name 'audit-enforce-jobs-on-automation-hybrid-runbook-workers' --
display-name 'Audit Enforce Jobs on Automation Hybrid Runbook Workers' --description 'This policy
enforces job execution on Automation account user Hybrid Runbook Workers.' --rules
'AuditAutomationHRWJobExecution.json' --mode All

```

The command creates a policy definition named **Audit Enforce Jobs on Automation Hybrid Runbook Workers**. For more information about other parameters that you can use, see [az policy definition create](#).

When called without location parameters, `az policy definition create` defaults to saving the policy definition in the selected subscription of the sessions context. To save the definition to a different location, use the following parameters:

- **subscription** - Save to a different subscription. Requires a *GUID* value for the subscription ID or a *string* value for the subscription name.
- **management-group** - Save to a management group. Requires a *string* value.

3. After you create your policy definition, you can create a policy assignment by running the following commands:

- [Azure CLI](#)
- [PowerShell](#)

```
az policy assignment create --name '<name>' --scope '<scope>' --policy '<policy definition ID>'
```

The **scope** parameter on `az policy assignment create` works with management group, subscription, resource group, or a single resource. The parameter uses a full resource path. The pattern for **scope** for each container is as follows. Replace `{rName}`, `{rgName}`, `{subID}`, and `{mgName}` with your resource name, resource group name, subscription ID, and management group name, respectively. `{rType}` would be replaced with the **resource type** of the resource, such as `Microsoft.Compute/virtualMachines` for a VM.

- Resource - `/subscriptions/{subID}/resourceGroups/{rgName}/providers/{rType}/{rName}`
- Resource group - `/subscriptions/{subID}/resourceGroups/{rgName}`
- Subscription - `/subscriptions/{subID}`
- Management group - `/providers/Microsoft.Management/managementGroups/{mgName}`

You can get the Azure Policy Definition ID by using PowerShell with the following command:

```
az policy definition show --name 'Audit Enforce Jobs on Automation Hybrid Runbook Workers'
```

The policy definition ID for the policy definition that you created should resemble the following example:

```
"/subscription/<subscriptionId>/providers/Microsoft.Authorization/policyDefinitions/Audit Enforce Jobs on Automation Hybrid Runbook Workers"
```

4. Sign in to the [Azure portal](#).
5. Launch the Azure Policy service in the Azure portal by selecting **All services**, then searching for and selecting **Policy**.
6. Select **Compliance** in the left side of the page. Then locate the policy assignment you created.

The screenshot shows the Microsoft Azure Policy Overview blade. At the top, there's a search bar and a scope filter set to 'Contoso'. The main area displays an overall resource compliance of 6% (9 out of 143) with 143 total resources. A donut chart indicates 9 Compliant, 0 Exempt, and 134 Non-compliant resources. Below this, there are sections for 'Non-compliant initiatives' (2 out of 3) and 'Non-compliant policies' (55 out of 212). A 'Related Services' sidebar lists 'Blueprints (preview)', 'Resource Graph', and 'User privacy'. On the right, a table provides detailed information for five specific resources:

Name	Scope	Compliance state	Resource compliant	Non-Compliant Resources	Non-compliant policies
ASC Default (subscription:75475e1e-96...)	Contoso	Non-compliant	6% (9 out of 143)	134	51
[Preview]: Enable Azure Monitor for VMs	IT Organization Mgmt Group	Non-compliant	58% (7 out of 12)	5	4
ASC DataProtection (subscription:75475e1e-96...)	Contoso	Compliant	100% (1 out of 1)	0	0
Enforce Jobs on Automation Hybrid Ru...	Contoso/MAIC-RG	Compliant	100% (0 out of 0)	0	0
[Preview]: Log Analytics agent should b...	IT Organization Mgmt Group	Compliant	100% (2 out of 2)	0	0

When one of the Automation REST operations are executed without reference to a Hybrid Runbook Worker in the request body, a 403 response code is returned with an error similar to the following example indicating the operation attempted execution on an Azure sandbox:

```
{
  "error": {
    "code": "RequestDisallowedByPolicy",
    "target": "Start_VMS",
    "message": "Resource 'Start_VMS' was disallowed by policy. Policy identifiers: '[{"policyAssignment": [{"name": "Enforce Jobs on Automation Hybrid Runbook Workers", "id": "/subscriptions/75475e1e-9643-4f3d-859e-055f4c31b458/resourceGroups/MAIC-RG/providers/Microsoft.Authorization/policyAssignments/fd5e2cb3842d4eefbc857917"}, {"policyDefinition": [{"name": "Enforce Jobs on Automation Hybrid Runbook Workers", "id": "/subscriptions/75475e1e-9643-4f3d-859e-055f4c31b458/providers/Microsoft.Authorization/policyDefinitions/4fdffd35-fd9f-458e-9779-94fe33401bfc"}]}]'." ,
    "additionalInfo": [
      {
        "type": "PolicyViolation",
        "info": {
          "policyDefinitionDisplayName": "Enforce Jobs on Automation Hybrid Runbook Workers",
          "evaluationDetails": {
            "evaluatedExpressions": [
              {
                "result": "True",
                "expressionKind": "Field",
                "expression": "type",
                "path": "type",
                "expressionValue": "Microsoft.Automation/automationAccounts/jobs",
                "targetValue": "Microsoft.Automation/automationAccounts/jobs",
                "operator": "Equals"
              },
              {
                "result": "True",
                "expressionKind": "Value",
                "expression": "[length(field('Microsoft.Automation/automationAccounts/jobs/runOn'))]",
                "expressionValue": 0,
                "targetValue": 1,
                "operator": "Less"
              }
            ]
          },
          "policyDefinitionId": "/subscriptions/75475e1e-9643-4f3d-859e-055f4c31b458/providers/Microsoft.Authorization/policyDefinitions/4fdffd35-fd9f-458e-9779-94fe33401bfc",
          "policyDefinitionName": "4fdffd35-fd9f-458e-9779-94fe33401bfc",
          "policyDefinitionEffect": "Deny",
          "policyAssignmentId": "/subscriptions/75475e1e-9643-4f3d-859e-055f4c31b458/resourceGroups/MAIC-RG/providers/Microsoft.Authorization/policyAssignments/fd5e2cb3842d4eefbc857917",
          "policyAssignmentName": "fd5e2cb3842d4eefbc857917",
          "policyAssignmentDisplayName": "Enforce Jobs on Automation Hybrid Runbook Workers",
          "policyAssignmentScope": "/subscriptions/75475e1e-9643-4f3d-859e-055f4c31b458/resourceGroups/MAIC-RG",
          "policyAssignmentParameters": {}
        }
      }
    ]
  }
}
```

The attempted operation is also logged in the Automation account's Activity Log, similar to the following example.

MAIC-AA-Pri | Activity log X

Automation Account

Search (Ctrl+/...)

Overview

Activity log (selected)

Access control (IAM)

Tags

Diagnose and solve problems

Configuration Management

Inventory

Change tracking

State configuration (DSC)

Update management

Update management

Activity Edit columns Refresh Diagnostics settings Download as CSV Logs Pin current filters Reset filters

Management Group : **None** Subscription : **Contoso** Event severity : **All** Timespan : **Last 6 hours** Resource group : **maic-rg**

Resource : **MAIC-AA-Pri** X Add Filter

6 items.

Operation name	Status	Time	Time stamp	Subscription	Event initiated by
1 'deny' Policy action.	Failed	11 minutes ...	Mon May 2...	Contoso	ssmith@contoso.com
1 Create an Azure Automation job	Started	11 minutes ...	Mon May 2...	Contoso	ssmith@contoso.com
1 Create an Azure Automation job	Started	11 minutes ...	Mon May 2...	Contoso	ssmith@contoso.com
1 Create an Azure Automation job	Failed	11 minutes ...	Mon May 2...	Contoso	ssmith@contoso.com
1 'deny' Policy action.	Failed	11 minutes ...	Mon May 2...	Contoso	ssmith@contoso.com
1 Create an Azure Automation job	Failed	11 minutes ...	Mon May 2...	Contoso	ssmith@contoso.com

Next steps

To work with runbooks, see [Manage runbooks in Azure Automation](#).

Troubleshoot Hybrid Runbook Worker issues

4/27/2021 • 13 minutes to read • [Edit Online](#)

This article provides information on troubleshooting and resolving issues with Azure Automation Hybrid Runbook Workers. For general information, see [Hybrid Runbook Worker overview](#).

General

The Hybrid Runbook Worker depends on an agent to communicate with your Azure Automation account to register the worker, receive runbook jobs, and report status. For Windows, this agent is the Log Analytics agent for Windows. For Linux, it's the Log Analytics agent for Linux.

Scenario: Runbook execution fails

Issue

Runbook execution fails, and you receive the following error message:

The job action 'Activate' cannot be run, because the process stopped unexpectedly. The job action was attempted three times.

Your runbook is suspended shortly after it attempts to execute three times. There are conditions that can interrupt the runbook from completing. The related error message might not include any additional information.

Cause

The following are possible causes:

- The runbooks can't authenticate with local resources.
- The hybrid worker is behind a proxy or firewall.
- The computer configured to run the Hybrid Runbook Worker doesn't meet the minimum hardware requirements.

Resolution

Verify that the computer has outbound access to *.azure-automation.net on port 443.

Computers running the Hybrid Runbook Worker should meet the minimum hardware requirements before the worker is configured to host this feature. Runbooks and the background process they use might cause the system to be overused and cause runbook job delays or timeouts.

Confirm that the computer to run the Hybrid Runbook Worker feature meets the minimum hardware requirements. If it does, monitor CPU and memory use to determine any correlation between the performance of Hybrid Runbook Worker processes and Windows. Any memory or CPU pressure can indicate the need to upgrade resources. You can also select a different compute resource that supports the minimum requirements and scale when workload demands indicate that an increase is necessary.

Check the **Microsoft-SMA** event log for a corresponding event with the description

Win32 Process Exited with code [4294967295]. The cause of this error is that you haven't configured authentication in your runbooks or specified the Run As credentials for the Hybrid Runbook Worker group. Review runbook permissions in [Running runbooks on a Hybrid Runbook Worker](#) to confirm that you've correctly configured authentication for your runbooks.

Scenario: Event 15011 in the Hybrid Runbook Worker

Issue

The Hybrid Runbook Worker receives event 15011, indicating that a query result isn't valid. The following error appears when the worker attempts to open a connection with the [SignalR server](#).

```
[AccountId={c7d22bd3-47b2-4144-bf88-97940102f6ca}] [Uri=https://cc-jobruntimedata-prod-su1.azure-automation.net/notifications/hub][Exception=System.TimeoutException: Transport timed out trying to connect at System.Runtime.ExceptionServices.ExceptionDispatchInfo.Throw() at System.Runtime.CompilerServices.TaskAwaiter.HandleNonSuccessAndDebuggerNotification(Task task) at JobRuntimeData.NotificationsClient.JobRuntimeDataServiceSignalRClient.<Start>d__45.MoveNext()]
```

Cause

The Hybrid Runbook Worker hasn't been configured correctly for the automated feature deployment, for example, for Update Management. The deployment contains a part that connects the VM to the Log Analytics workspace. The PowerShell script looks for the workspace in the subscription with the supplied name. In this case, the Log Analytics workspace is in a different subscription. The script can't find the workspace and tries to create one, but the name is already taken. As a result, the deployment fails.

Resolution

You have two options for resolving this issue:

- Modify the PowerShell script to look for the Log Analytics workspace in another subscription. This is a good resolution to use if you plan to deploy many Hybrid Runbook Worker machines in the future.
- Manually configure the worker machine to run in an Orchestrator sandbox. Then run a runbook created in the Azure Automation account on the worker to test the functionality.

Scenario: Windows Azure VMs automatically dropped from a hybrid worker group

Issue

You can't see the Hybrid Runbook Worker or VMs when the worker machine has been turned off for a long time.

Cause

The Hybrid Runbook Worker machine hasn't pinged Azure Automation for more than 30 days. As a result, Automation has purged the Hybrid Runbook Worker group or the System Worker group.

Resolution

Start the worker machine, and then rereregister it with Azure Automation. For instructions on how to install the runbook environment and connect to Azure Automation, see [Deploy a Windows Hybrid Runbook Worker](#).

Scenario: No certificate was found in the certificate store on the Hybrid Runbook Worker

Issue

A runbook running on a Hybrid Runbook Worker fails with the following error message:

```
Connect-AzAccount : No certificate was found in the certificate store with thumbprint  
000000000000000000000000000000000000000000000000000000000000000  
  
At line:3 char:1  
+ Connect-AzAccount -ServicePrincipal -Tenant $Conn.TenantID -Appl ...  
+ ~~~~~  
+ CategoryInfo : CloseError: (:) [Connect-AzAccount], ArgumentException  
+ FullyQualifiedErrorId : Microsoft.Azure.Commands.Profile.ConnectAzAccountCommand
```

Cause

This error occurs when you attempt to use a [Run As account](#) in a runbook that runs on a Hybrid Runbook Worker where the Run As account certificate isn't present. Hybrid Runbook Workers don't have the certificate asset locally by default. The Run As account requires this asset to operate properly.

Resolution

If your Hybrid Runbook Worker is an Azure VM, you can use [runbook authentication with managed identities](#) instead. This scenario simplifies authentication by allowing you to authenticate to Azure resources using the managed identity of the Azure VM instead of the Run As account. When the Hybrid Runbook Worker is an on-premises machine, you need to install the Run As account certificate on the machine. To learn how to install the certificate, see the steps to run the PowerShell runbook [Export-RunAsCertificateToHybridWorker](#) in [Run runbooks on a Hybrid Runbook Worker](#).

Scenario: Error 403 during registration of a Hybrid Runbook Worker

Issue

The worker's initial registration phase fails, and you receive the following error (403):

```
Forbidden: You don't have permission to access / on this server.
```

Cause

The following issues are possible causes:

- There's a mistyped workspace ID or workspace key (primary) in the agent's settings.
- The Hybrid Runbook Worker can't download the configuration, which causes an account linking error. When Azure enables features on machines, it supports only certain regions for linking a Log Analytics workspace and an Automation account. It's also possible that an incorrect date or time is set on the computer. If the time is +/- 15 minutes from the current time, feature deployment fails.

Resolution

Mistyped workspace ID or key

To verify if the agent's workspace ID or workspace key was mistyped, see [Adding or removing a workspace - Windows agent](#) for the Windows agent or [Adding or removing a workspace - Linux agent](#) for the Linux agent. Make sure to select the full string from the Azure portal, and copy and paste it carefully.

Configuration not downloaded

Your Log Analytics workspace and Automation account must be in a linked region. For a list of supported regions, see [Azure Automation and Log Analytics workspace mappings](#).

You might also need to update the date or time zone of your computer. If you select a custom time range, make sure that the range is in UTC, which can differ from your local time zone.

Scenario: Set-AzStorageBlobContent fails on a Hybrid Runbook Worker

Issue

Runbook fails when it tries to execute `Set-AzStorageBlobContent`, and you receive the following error message:

```
Set-AzStorageBlobContent : Failed to open filexxxxxxxxxxxxxx: Illegal characters in path
```

Cause

This error is caused by the long file name behavior of calls to `[System.IO.Path]::GetFullPath()` which adds UNC paths.

Resolution

As a workaround, you can create a configuration file named `OrchestratorSandbox.exe.config` with the following content:

```
<configuration>
  <runtime>
    <AppContextSwitchOverrides value="Switch.System.IO.UseLegacyPathHandling=false" />
  </runtime>
</configuration>
```

Place this file in the same folder as the executable file `OrchestratorSandbox.exe`. For example,

```
%ProgramFiles%\Microsoft Monitoring Agent\Agent\AzureAutomation\7.3.702.0\HybridAgent
```

NOTE

If you upgrade the agent, this config file will be deleted, and will need to be recreated.

Linux

The Linux Hybrid Runbook Worker depends on the [Log Analytics agent for Linux](#) to communicate with your

Automation account to register the worker, receive runbook jobs, and report status. If registration of the worker fails, here are some possible causes for the error.

Scenario: Linux Hybrid Runbook Worker receives prompt for a password when signing a runbook

Issue

Running the `sudo` command for a Linux Hybrid Runbook Worker retrieves an unexpected prompt for a password.

Cause

The `nxautomationuser` account for the Log Analytics agent for Linux is not correctly configured in the `sudoers` file. The Hybrid Runbook Worker needs the appropriate configuration of account permissions and other data so that it can sign runbooks on the Linux Runbook Worker.

Resolution

- Ensure that the Hybrid Runbook Worker has the GnuPG (GPG) executable on the machine.
- Verify the configuration of the `nxautomationuser` account in the `sudoers` file. See [Running runbooks on a Hybrid Runbook Worker](#).

Scenario: Log Analytics agent for Linux isn't running

Issue

The Log Analytics agent for Linux isn't running.

Cause

If the agent isn't running, it prevents the Linux Hybrid Runbook Worker from communicating with Azure Automation. The agent might not be running for various reasons.

Resolution

Verify the agent is running by entering the command `ps -ef | grep python`. You should see output similar to the following. The Python processes with the `nxautomation` user account. If the Azure Automation feature isn't enabled, none of the following processes are running.

```
nxautom+ 8567      1  0 14:45 ?      00:00:00 python
/opt/microsoft/omsconfig/modules/nxOMSAutomationWorker/DSCResources/MSFT_nxOMSAutomationWorkerResource/automationworker/worker/main.py /var/opt/microsoft/omsagent/state/automationworker/oms.conf rworkspace:<workspaceId> <Linux hybrid worker version>
nxautom+ 8593      1  0 14:45 ?      00:00:02 python
/opt/microsoft/omsconfig/modules/nxOMSAutomationWorker/DSCResources/MSFT_nxOMSAutomationWorkerResource/automationworker/worker/hybridworker.py /var/opt/microsoft/omsagent/state/automationworker/worker.conf managed rworkspace:<workspaceId> rversion:<Linux hybrid worker version>
nxautom+ 8595      1  0 14:45 ?      00:00:02 python
/opt/microsoft/omsconfig/modules/nxOMSAutomationWorker/DSCResources/MSFT_nxOMSAutomationWorkerResource/automationworker/worker/hybridworker.py
/var/opt/microsoft/omsagent/<workspaceId>/state/automationworker/diy/worker.conf managed rworkspace:<workspaceId> rversion:<Linux hybrid worker version>
```

The following list shows the processes that are started for a Linux Hybrid Runbook Worker. They're all located in the `/var/opt/microsoft/omsagent/state/automationworker/` directory.

- **oms.conf**: The worker manager process. It's started directly from DSC.
- **worker.conf**: The Auto-Registered hybrid worker process. It's started by the worker manager. This process is used by Update Management and is transparent to the user. This process isn't present if Update Management isn't enabled on the machine.
- **diy/worker.conf**: The DIY hybrid worker process. The DIY hybrid worker process is used to execute user runbooks on the Hybrid Runbook Worker. It only differs from the Auto-registered hybrid worker process in the key detail that it uses a different configuration. This process isn't present if Azure Automation is disabled and the DIY Linux Hybrid Worker isn't registered.

If the agent isn't running, run the following command to start the service:

```
sudo /opt/microsoft/omsagent/bin/service_control restart .
```

Scenario: The specified class doesn't exist

If you see the error message `The specified class does not exist..` in `/var/opt/microsoft/omsconfig/omsconfig.log`, the Log Analytics agent for Linux needs to be updated. Run the following command to reinstall the agent.

```
wget https://raw.githubusercontent.com/Microsoft/OMS-Agent-for-Linux/master/installer/scripts/onboard_agent.sh && sh onboard_agent.sh -w <WorkspaceID> -s <WorkspaceKey>
```

Windows

The Windows Hybrid Runbook Worker depends on the [Log Analytics agent for Windows](#) to communicate with your Automation account to register the worker, receive runbook jobs, and report status. If registration of the worker fails, this section includes some possible reasons.

Scenario: The Log Analytics agent for Windows isn't running

Issue

The `healthservice` isn't running on the Hybrid Runbook Worker machine.

Cause

If the Log Analytics for Windows service isn't running, the Hybrid Runbook Worker can't communicate with Azure Automation.

Resolution

Verify that the agent is running by entering the following command in PowerShell: `Get-Service healthservice`. If the service is stopped, enter the following command in PowerShell to start the service:

```
Start-Service healthservice .
```

Scenario: Event 4502 in the Operations Manager log

Issue

In the Application and Services Logs\Operations Manager event log, you see event 4502 and an event message that contains `Microsoft.EnterpriseManagement.HealthService.AzureAutomation.HybridAgent` with the following description:

```
The certificate presented by the service \<wsid\>.oms.opinsights.azure.com was not issued by a certificate authority used for Microsoft services. Please contact your network administrator to see if they are running a proxy that intercepts TLS/SSL communication.
```

Cause

This issue can be caused by your proxy or network firewall blocking communication to Microsoft Azure. Verify that the computer has outbound access to `*.azure-automation.net` on port 443.

Resolution

Logs are stored locally on each hybrid worker at `C:\ProgramData\Microsoft\System Center\Orchestrator\7.2\SMA\Sandboxes`. You can verify if there are any warning or error events in the **Application and Services Logs\Microsoft-SMA\Operations and Application and Services Logs\Operations Manager** event logs. These logs indicate a connectivity or other type of issue that affects the enabling of the role to Azure Automation, or an issue encountered under normal operations. For additional help troubleshooting issues with the Log Analytics agent, see [Troubleshoot issues with the Log Analytics Windows agent](#).

Hybrid workers send [Runbook output and messages](#) to Azure Automation in the same way that runbook jobs running in the cloud send output and messages. You can enable the Verbose and Progress streams just as you do for runbooks.

Scenario: Orchestrator.Sandbox.exe can't connect to Microsoft 365 through proxy

Issue

A script running on a Windows Hybrid Runbook Worker can't connect as expected to Microsoft 365 on an Orchestrator sandbox. The script is using [Connect-MsolService](#) for connection.

If you adjust **Orchestrator.Sandbox.exe.config** to set the proxy and the bypass list, the sandbox still doesn't connect properly. A **Powershell_ise.exe.config** file with the same proxy and bypass list settings seems to work as you expect. Service Management Automation (SMA) logs and PowerShell logs don't provide any information regarding proxy.

Cause

The connection to Active Directory Federation Services (AD FS) on the server can't bypass the proxy. Remember that a PowerShell sandbox runs as the logged user. However, an Orchestrator sandbox is heavily customized and might ignore the **Orchestrator.Sandbox.exe.config** file settings. It has special code for handling machine or Log Analytics agent proxy settings, but not for handling other custom proxy settings.

Resolution

You can resolve the issue for the Orchestrator sandbox by migrating your script to use the Azure Active Directory modules instead of the MSOnline module for PowerShell cmdlets. For more information, see [Migrating from Orchestrator to Azure Automation \(Beta\)](#).

If you want to continue to use the MSOnline module cmdlets, change your script to use [Invoke-Command](#).

Specify values for the `ComputerName` and `Credential` parameters.

```
$Credential = Get-AutomationPSCredential -Name MyProxyAccessibleCredential  
Invoke-Command -ComputerName $env:COMPUTERNAME -Credential $Credential  
{ Connect-MsolService ... }
```

This code change starts an entirely new PowerShell session under the context of the specified credentials. It should enable the traffic to flow through a proxy server that's authenticating the active user.

NOTE

This resolution makes it unnecessary to manipulate the sandbox configuration file. Even if you succeed in making the configuration file work with your script, the file gets wiped out each time the Hybrid Runbook Worker agent is updated.

Scenario: Hybrid Runbook Worker not reporting

Issue

Your Hybrid Runbook Worker machine is running, but you don't see heartbeat data for the machine in the workspace.

The following example query shows the machines in a workspace and their last heartbeat:

```
Heartbeat  
| summarize arg_max(TimeGenerated, *) by Computer
```

Cause

This issue can be caused by a corrupt cache on the Hybrid Runbook Worker.

Resolution

To resolve this issue, sign in to the Hybrid Runbook Worker and run the following script. This script stops the Log Analytics agent for Windows, removes its cache, and restarts the service. This action forces the Hybrid Runbook Worker to re-download its configuration from Azure Automation.

```
Stop-Service -Name HealthService  
  
Remove-Item -Path 'C:\Program Files\Microsoft Monitoring Agent\Agent\Health Service State' -Recurse  
  
Start-Service -Name HealthService
```

Scenario: You can't add a Windows Hybrid Runbook Worker

Issue

You receive the following message when you try to add a Hybrid Runbook Worker by using the `Add-HybridRunbookWorker` cmdlet:

```
Machine is already registered
```

Cause

This issue can be caused if the machine is already registered with a different Automation account or if you try to re-add the Hybrid Runbook Worker after removing it from a machine.

Resolution

To resolve this issue, remove the following registry key, restart `HealthService`, and try the `Add-HybridRunbookWorker` cmdlet again.

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\HybridRunbookWorker
```

Scenario: You can't add a Linux Hybrid Runbook Worker

Issue

You receive the following message when you try to add a Hybrid Runbook Worker by using the `sudo python /opt/microsoft/omsconfig/.../onboarding.py --register` python script:

```
Unable to register, an existing worker was found. Please deregister any existing worker and try again.
```

Additionally, attempting to deregister a Hybrid Runbook Worker by using the `sudo python /opt/microsoft/omsconfig/.../onboarding.py --deregister` python script:

```
Failed to deregister worker. [response_status=404]
```

Cause

This issue might occur if the machine is already registered with a different Automation account, if the Azure Hybrid Worker Group was deleted, or if you try to re-add the Hybrid Runbook Worker after you remove it from a machine.

Resolution

To resolve this issue:

1. Remove the agent `sudo sh onboard_agent.sh --purge`.

2. Run these commands:

```
sudo mv -f /home/nxautomation/state/worker.conf /home/nxautomation/state/worker.conf_old  
sudo mv -f /home/nxautomation/state/worker_diy.crt /home/nxautomation/state/worker_diy.crt_old  
sudo mv -f /home/nxautomation/state/worker_diy.key /home/nxautomation/state/worker_diy.key_old
```

3. Re-onboard the agent

```
sudo sh onboard_agent.sh -w <workspace id> -s <workspace key> -d opinsights.azure.com .
```

4. Wait for the folder `/opt/microsoft/omsconfig/modules/nxOMSAutomationWorker` to populate.

5. Try the `sudo python /opt/microsoft/omsconfig/.../onboarding.py --register` Python script again.

Next steps

If you don't see your problem here or you can't resolve your issue, try one of the following channels for additional support:

- Get answers from Azure experts through [Azure Forums](#).
- Connect with [@AzureSupport](#), the official Microsoft Azure account for improving customer experience. Azure Support connects the Azure community to answers, support, and experts.
- File an Azure support incident. Go to the [Azure support site](#), and select **Get Support**.

Use source control integration

4/22/2021 • 6 minutes to read • [Edit Online](#)

Source control integration in Azure Automation supports single-direction synchronization from your source control repository. Source control allows you to keep your runbooks in your Automation account up to date with scripts in your GitHub or Azure Repos source control repository. This feature makes it easy to promote code that has been tested in your development environment to your production Automation account.

Source control integration lets you easily collaborate with your team, track changes, and roll back to earlier versions of your runbooks. For example, source control allows you to synchronize different branches in source control with your development, test, and production Automation accounts.

Source control types

Azure Automation supports three types of source control:

- GitHub
- Azure Repos (Git)
- Azure Repos (TFVC)

Prerequisites

- A source control repository (GitHub or Azure Repos)
- A [Run As account](#)
- The `AzureRM.Profile` module must be imported into your Automation account. Note that the equivalent Az module (`Az.Accounts`) will not work with Automation source control.

NOTE

Source control synchronization jobs are run under the user's Automation account and are billed at the same rate as other Automation jobs.

Configure source control

This section tells how to configure source control for your Automation account. You can use either the Azure portal or PowerShell.

Configure source control in Azure portal

Use this procedure to configure source control using the Azure portal.

1. In your Automation account, select **Source Control** and click **Add**.

The screenshot shows the Azure portal interface for an Automation Account named 'ContosoFinance'. The left sidebar has a tree view with 'Source control' selected. The main content area has a heading 'Source control' with a sub-section 'Source controls'. Below this is a table with columns: SOURCE TYPE, NAME, CREATED, LAST MODIFIED, and AUTO SYNC. A single row 'No results' is listed.

2. Choose **Source Control type**, then click **Authenticate**.
3. A browser window opens and prompts you to sign in. Follow the prompts to complete authentication.
4. On the Source Control Summary page, use the fields to fill in the source control properties defined below. Click **Save** when finished.

PROPERTY	DESCRIPTION
Source control name	A friendly name for the source control. This name must contain only letters and numbers.
Source control type	Type of source control mechanism. Available options are: * GitHub * Azure Repos (Git) * Azure Repos (TFVC)
Repository	Name of the repository or project. The first 200 repositories are retrieved. To search for a repository, type the name in the field and click Search on GitHub .
Branch	Branch from which to pull the source files. Branch targeting isn't available for the TFVC source control type.
Folder path	Folder that contains the runbooks to synchronize, for example, <code>/Runbooks</code> . Only runbooks in the specified folder are synchronized. Recursion isn't supported.
Auto Sync ¹	Setting that turns on or off automatic synchronization when a commit is made in the source control repository.
Publish Runbook	Setting of On if runbooks are automatically published after synchronization from source control, and Off otherwise.
Description	Text specifying additional details about the source control.

¹ To enable Auto Sync when configuring source control integration with Azure Repos, you must be a Project Administrator.

Source Control Summary

Start Sync Save Delete Discard

* Source control name
ContosoFinanceRunbooks

* Source control type
Azure Repos (TFVC)

Authorization successful.

* Repository
\$/ContosoFinanceTFVCExample

Branch
Branch is not supported for VsoTfvc.

* Folder path
/Runbooks

Auto Sync
 On Off

Publish Runbook
 On Off

Description

NOTE

The login for your source control repository might be different from your login for the Azure portal. Ensure that you are logged in with the correct account for your source control repository when configuring source control. If there is a doubt, open a new tab in your browser, log out from [dev.azure.com](#), [visualstudio.com](#), or [github.com](#), and try reconnecting to source control.

Configure source control in PowerShell

You can also use PowerShell to configure source control in Azure Automation. To use the PowerShell cmdlets for this operation, you need a personal access token (PAT). Use the [New-AzAutomationSourceControl](#) cmdlet to create the source control connection. This cmdlet requires a secure string for the PAT. To learn how to create a secure string, see [ConvertTo-SecureString](#).

The following subsections illustrate PowerShell creation of the source control connection for GitHub, Azure Repos (Git), and Azure Repos (TFVC).

Create source control connection for GitHub

```
New-AzAutomationSourceControl -Name SCGitHub -RepoUrl https://github.com/<accountname>/<reponame>.git -  
SourceType GitHub -FolderPath "/MyRunbooks" -Branch master -AccessToken <secureStringofPAT> -  
ResourceGroupName <ResourceGroupName> -AutomationAccountName <AutomationAccountName>
```

Create source control connection for Azure Repos (Git)

NOTE

Azure Repos (Git) uses a URL that accesses `dev.azure.com` instead of `visualstudio.com`, used in earlier formats. The older URL format `https://<accountname>.visualstudio.com/<projectname>/_git/<repositoryname>` is deprecated but still supported. The new format is preferred.

```
New-AzAutomationSourceControl -Name SCReposGit -RepoUrl  
https://dev.azure.com/<accountname>/<adoprojectname>/_git/<repositoryname> -SourceType VsoGit -AccessToken  
<secureStringofPAT> -Branch master -ResourceGroupName <ResourceGroupName> -AutomationAccountName  
<AutomationAccountName> -FolderPath "/Runbooks"
```

Create source control connection for Azure Repos (TFVC)

NOTE

Azure Repos (TFVC) uses a URL that accesses `dev.azure.com` instead of `visualstudio.com`, used in earlier formats. The older URL format `https://<accountname>.visualstudio.com/<projectname>/_versionControl` is deprecated but still supported. The new format is preferred.

```
New-AzAutomationSourceControl -Name SCRepostFV -RepoUrl  
https://dev.azure.com/<accountname>/<adoprojectname>/_git/<repositoryname> -SourceType VsoTfvc -AccessToken  
<secureStringofPAT> -ResourceGroupName <ResourceGroupName> -AutomationAccountName <AutomationAccountName> -  
FolderPath "/Runbooks"
```

Personal access token (PAT) permissions

Source control requires some minimum permissions for PATs. The following subsections contain the minimum permissions required for GitHub and Azure Repos.

Minimum PAT permissions for GitHub

The following table defines the minimum PAT permissions required for GitHub. For more information about creating a PAT in GitHub, see [Creating a personal access token for the command line](#).

SCOPE	DESCRIPTION
<code>repo</code>	
<code>repo:status</code>	Access commit status
<code>repo_deployment</code>	Access deployment status
<code>public_repo</code>	Access public repositories
<code>repo:invite</code>	Access repository invitations
<code>security_events</code>	Read and write security events
<code>admin:repo_hook</code>	
<code>write:repo_hook</code>	Write repository hooks
<code>read:repo_hook</code>	Read repository hooks

The following list defines the minimum PAT permissions required for Azure Repos. For more information about creating a PAT in Azure Repos, see [Authenticate access with personal access tokens](#).

SCOPE	ACCESS TYPE
Code	Read
Project and team	Read
Identity	Read
User profile	Read
Work items	Read
Service connections	Read, query, manage ¹

¹ The Service connections permission is only required if you have enabled autosync.

Synchronize with source control

Follow these steps to synchronize with source control.

1. Select the source from the table on the Source control page.
2. Click **Start Sync** to start the sync process.
3. View the status of the current sync job or previous ones by clicking the **Sync jobs** tab.
4. On the **Source Control** dropdown menu, select a source control mechanism.

The screenshot shows the 'ContosoFinance - Source control' page. The left sidebar has a tree view with 'Account Settings' expanded, showing 'Properties', 'Keys', 'Pricing', and 'Source control' (which is selected). Other options include 'Run as accounts', 'Settings', 'Locks', 'Automation script', 'Support + troubleshooting', and 'New support request'. The main area has a search bar and 'Add' and 'Refresh' buttons. A descriptive text block explains that source control allows runbooks to be up-to-date with scripts in an external source control. It includes links to 'Learn more', 'Source Control', and 'Provide feedback'. Below this is a 'Source controls' section with a table showing three sync jobs:

STATUS	SOURCE CONTROL	SYNC TYPE	CREATION TIME	END TIME
✓ Completed	ContosoFinanceRunbooks	Full	9/17/2018, 12:11 PM	9/17/2018, 12:12 PM
✓ Completed	ContosoFinanceRunbooks	Full	9/17/2018, 11:28 AM	9/17/2018, 11:29 AM
✗ Failed	ContosoFinanceRunbooks	Full	9/17/2018, 11:24 AM	9/17/2018, 11:24 AM

5. Clicking on a job allows you to view the job output. The following example is the output from a source control sync job.

```
=====
Azure Automation Source Control.
Supported runbooks to sync: PowerShell Workflow, PowerShell Scripts, DSC Configurations, Graphical, and Python 2.

Setting AzEnvironment.

Getting AzureRunAsConnection.

Logging in to Azure...

Source control information for syncing:

[Url = https://ContosoExample.visualstudio.com/ContosoFinanceTFVCExample/_versionControl] [FolderPath = /Runbooks]

Verifying url: https://ContosoExample.visualstudio.com/ContosoFinanceTFVCExample/_versionControl

Connecting to VSTS...

Source Control Sync Summary:

2 files synced:
- ExampleRunbook1.ps1
- ExampleRunbook2.ps1
```

6. Additional logging is available by selecting **All Logs** on the Source Control Sync Job Summary page. These additional log entries can help you troubleshoot issues that might arise when using source control.

Disconnect source control

To disconnect from a source control repository:

1. Open **Source control** under **Account Settings** in your Automation account.
2. Select the source control mechanism to remove.
3. On the Source Control Summary page, click **Delete**.

Handle encoding issues

If multiple people are editing runbooks in your source control repository using different editors, encoding issues can occur. To learn more about this situation, see [Common causes of encoding issues](#).

Update the PAT

Currently, you can't use the Azure portal to update the PAT in source control. When your PAT is expired or revoked, you can update source control with a new access token in one of these ways:

- Use the [REST API](#).
- Use the [Update-AzAutomationSourceControl](#) cmdlet.

Next steps

- For integrating source control in Azure Automation, see [Azure Automation: Source Control Integration in Azure Automation](#).

- For integrating runbook source control with Visual Studio Codespaces, see [Azure Automation: Integrating Runbook Source Control using Visual Studio Codespaces](#).

Azure Automation State Configuration overview

8/18/2021 • 4 minutes to read • [Edit Online](#)

Azure Automation State Configuration is an Azure configuration management service that allows you to write, manage, and compile PowerShell Desired State Configuration (DSC) [configurations](#) for nodes in any cloud or on-premises datacenter. The service also imports [DSC Resources](#), and assigns configurations to target nodes, all in the cloud. You can access Azure Automation State Configuration in the Azure portal by selecting **State configuration (DSC)** under **Configuration Management**.

NOTE

Before you enable Automation State Configuration, we would like you to know about [Azure Policy guest configuration](#), which can audit or configure settings inside machines running in Azure or on machines outside of Azure connected with [Arc-enabled servers](#). Azure Policy guest configuration offers similar functionality and is designed to take advantage of newer technology in Azure.

You can use Azure Automation State Configuration to manage a variety of machines:

- Azure virtual machines
- Azure virtual machines (classic)
- Physical/virtual Windows machines on-premises, or in a cloud other than Azure (including AWS EC2 instances)
- Physical/virtual Linux machines on-premises, in Azure, or in a cloud other than Azure

If you aren't ready to manage machine configuration from the cloud, you can use Azure Automation State Configuration as a report-only endpoint. This feature allows you to set (push) configurations through DSC and view reporting details in Azure Automation.

NOTE

Managing Azure VMs with Azure Automation State Configuration is included at no extra charge if the installed Azure VM Desired State Configuration extension version is greater than 2.70. For more information, see [Automation pricing page](#).

Why use Azure Automation State Configuration

Azure Automation State Configuration provides several advantages over the use of DSC outside of Azure. This service enables scalability across thousands of machines quickly and easily from a central, secure location. You can easily enable machines, assign them declarative configurations, and view reports showing each machine's compliance with the desired state you specify.

The Azure Automation State Configuration service is to DSC what Azure Automation runbooks are to PowerShell scripting. In other words, in the same way that Azure Automation helps you manage PowerShell scripts, it also helps you manage DSC configurations.

Built-in pull server

Azure Automation State Configuration provides a DSC pull server similar to the [Windows Feature DSC-Service](#). Target nodes can automatically receive configurations, conform to the desired state, and report on their compliance. The built-in pull server in Azure Automation eliminates the need to set up and maintain your own

pull server. Azure Automation can target virtual or physical Windows or Linux machines, in the cloud or on-premises.

Management of all your DSC artifacts

Azure Automation State Configuration brings the same management layer to [PowerShell Desired State Configuration](#) as it offers for PowerShell scripting. From the Azure portal or from PowerShell, you can manage all your DSC configurations, resources, and target nodes.

The screenshot shows the Azure portal interface for managing State Configuration. The left sidebar has a navigation path: Home > automation - State configuration (DSC). The main area is titled "automation - State configuration (DSC)" under "Automation Account". A search bar shows "State Config". There are buttons for "+ Add", "Refresh", "Reset filters", and "Enable Log Search". The top navigation tabs are "Nodes", "Configurations", "Compiled configurations", and "Gallery". The "Nodes" tab is selected. A large circular gauge indicates the status of configurations: 0 failed, 0 pending, 0 not completed, 0 in progress, 0 unresponsive, and 0 compiled. Below the gauge are filters for "Nodes", "Status", "Node configuration", and "VM DSC extension version". A table at the bottom lists columns: NODE, STATUS, NODE CONFIGURATION, LAST SEEN, and VERSION, with a note "No data". On the right, there's a link to "Learn more" and "Add non-Azure machine".

Import of reporting data into Azure Monitor logs

Nodes that are managed with Azure Automation State Configuration send detailed reporting status data to the built-in pull server. You can configure Azure Automation State Configuration to send this data to your Log Analytics workspace. See [Forward Azure Automation State Configuration reporting data to Azure Monitor logs](#).

Prerequisites

Consider the requirements in this section when using Azure Automation State Configuration.

Operating system requirements

For nodes running Windows, the following versions are supported:

- Windows Server 2019
- Windows Server 2016
- Windows Server 2012R2
- Windows Server 2012
- Windows Server 2008 R2 SP1
- Windows 10
- Windows 8.1
- Windows 7

NOTE

The [Microsoft Hyper-V Server](#) standalone product SKU does not contain an implementation of DSC. Thus it can't be managed by PowerShell DSC or Azure Automation State Configuration.

For nodes running Linux, the DSC Linux extension supports all the Linux distributions listed in the [PowerShell DSC documentation](#).

DSC requirements

For all Windows nodes running in Azure, [WMF 5.1](#) is installed when machines are enabled. For nodes running

Windows Server 2012 and Windows 7, [WinRM](#) is enabled.

For all Linux nodes running in Azure, [PowerShell DSC for Linux](#) is installed when machines are enabled.

Configuration of private networks

Check [Azure Automation Network Configuration](#) for detailed information on the ports, URLs, and other networking details required for nodes on a private network.

Proxy support

Proxy support for the DSC agent is available in Windows version 1809 and later. This option is enabled by setting the values for `ProxyURL` and `ProxyCredential` properties in the [metaconfiguration script](#) used to register nodes.

NOTE

Azure Automation State Configuration does not provide DSC proxy support for previous versions of Windows.

For Linux nodes, the DSC agent supports proxy and uses the `http_proxy` variable to determine the URL. To find out more about proxy support, see [Generate DSC metaconfigurations](#).

DNS records per region

It's recommended to use the addresses listed in the [DNS records per region](#) table when defining exceptions.

Next steps

- To get started, see [Get started with Azure Automation State Configuration](#).
- To learn how to enable nodes, see [Enable Azure Automation State Configuration](#).
- To learn about compiling DSC configurations so that you can assign them to target nodes, see [Compile DSC configurations in Azure Automation State Configuration](#).
- To see an example of using Azure Automation State Configuration in a continuous deployment pipeline, see [Set up continuous deployment with Chocolatey](#).
- For pricing information, see [Azure Automation State Configuration pricing](#).
- For a PowerShell cmdlet reference, see [Az.Automation](#).

Configure Linux desired state with Azure Automation State Configuration using PowerShell

9/1/2021 • 9 minutes to read • [Edit Online](#)

In this tutorial, you'll apply an Azure Automation State Configuration with PowerShell to an Azure Linux virtual machine to check whether it complies with a desired state. The desired state is to identify if the apache2 service is present on the node.

Azure Automation State Configuration allows you to specify configurations for your machines and ensure those machines are in a specified state over time. For more information about State Configuration, see [Azure Automation State Configuration overview](#).

In this tutorial, you learn how to:

- Onboard an Azure Linux VM to be managed by Azure Automation DSC
- Compose a configuration
- Install PowerShell module for Automation
- Import a configuration to Azure Automation
- Compile a configuration into a node configuration
- Assign a node configuration to a managed node
- Modify the node configuration mapping
- Check the compliance status of a managed node

If you don't have an Azure subscription, create a [free account](#) before you begin.

Prerequisites

- An Azure Automation account. To learn more about Automation accounts, see [Automation Account authentication overview](#).
- An Azure Resource Manager virtual machine (VM) running Ubuntu 18.04 LTS or later. For instructions on creating an Azure Linux VM, see [Create a Linux virtual machine in Azure with PowerShell](#).
- The PowerShell [Az Module](#) installed on the machine you'll be using to write, compile, and apply a state configuration to a target Azure Linux VM. Ensure you have the latest version. If necessary, run
`Update-Module -Name Az`.

Create a configuration

Review the code below and note the presence of two node configurations: `IsPresent` and `IsNotPresent`. This configuration calls one resource in each node block: the [nxPackage resource](#). This resource manages the presence of the [apache2](#) package. Then, in a text editor, copy the following code to a local file and name it `LinuxConfig.ps1`:

```

Configuration LinuxConfig
{
    Import-DscResource -ModuleName 'nx'

    Node IsPresent
    {
        nxPackage apache2
        {
            Name          = 'apache2'
            Ensure        = 'Present'
            PackageManager = 'Apt'
        }
    }

    Node IsNotPresent
    {
        nxPackage apache2
        {
            Name          = 'apache2'
            Ensure        = 'Absent'
        }
    }
}

```

Sign in to Azure

From your machine, sign in to your Azure subscription with the [Connect-AzAccount](#) PowerShell cmdlet and follow the on-screen directions.

```

# Sign in to your Azure subscription
$sub = Get-AzSubscription -ErrorAction SilentlyContinue
if(-not($sub))
{
    Connect-AzAccount
}

# If you have multiple subscriptions, set the one to use
# Select-AzSubscription -SubscriptionId "<SUBSCRIPTIONID>"
```

Initialize variables

For efficiency and decreased chance of error when executing the cmdlets, revise the PowerShell code further below as necessary and then execute.

VARIABLE	VALUE
\$resourceGroup	Replace <code>yourResourceGroup</code> with the actual name of your resource group.
\$automationAccount	Replace <code>yourAutomationAccount</code> with the actual name of your Automation account.
\$VM	Replace <code>yourVM</code> with the actual name of your Azure Linux VM.
\$configurationName	Leave as is with <code>LinuxConfig</code> . The name of the configuration used in this tutorial.

VARIABLE	VALUE
\$nodeConfigurationName0	Leave as is with <code>LinuxConfig.IsNotNullPresent</code> . The name of a node configuration used in this tutorial.
\$nodeConfigurationName1	Leave as is with <code>LinuxConfig.isPresent</code> . The name of a node configuration used in this tutorial.
\$moduleName	Leave as is with <code>nx</code> . The name of the PowerShell module used for DSC in this tutorial.
\$moduleVersion	Obtain the latest version number for <code>nx</code> from the PowerShell Gallery . This tutorial uses version <code>1.0</code> .

```
$resourceGroup = "yourResourceGroup"
$automationAccount = "yourAutomationAccount"
$VM = "yourVM"
$configurationName = "LinuxConfig"
$nodeConfigurationName0 = "LinuxConfig.IsNotNullPresent"
$nodeConfigurationName1 = "LinuxConfig.isPresent"
$moduleName = "nx"
$moduleVersion = "1.0"
```

Install nx module

Azure Automation uses a number of PowerShell modules to enable cmdlets in runbooks and DSC resources in DSC configurations. `nx` is the module with DSC Resources for Linux. Install the `nx` module with the [New-AzAutomationModule](#) cmdlet. For more information about modules, see [Manage modules in Azure Automation](#).

Run the following command:

```
New-AzAutomationModule ` 
-ResourceGroupName $resourceGroup ` 
-AutomationAccountName $automationAccount ` 
-Name $moduleName ` 
-ContentLinkUri "https://www.powershellgallery.com/api/v2/package/$moduleName/$moduleVersion"
```

The output should look similar as shown below:

```
ResourceGroupName      : ContosoLab
AutomationAccountName : MyAutomationAccount
Name                  : nx
IsGlobal              : False
Version               : 1.0
SizeInBytes           : 8045
ActivityCount         : 10
CreationTime          : 6/21/2021 9:16:26 PM +00:00
LastModifiedTime       : 6/21/2021 9:43:20 PM +00:00
ProvisioningState      : Creating
```

You can verify the installation running the following command:

```
Get-AzAutomationModule ` 
-ResourceGroupName $resourceGroup ` 
-AutomationAccountName $automationAccount ` 
-Name $moduleName
```

Import configuration to Azure Automation

Call the [Import-AzAutomationDscConfiguration](#) cmdlet to upload the configuration into your Automation account. Revise value for `-SourcePath` with your actual path and then run the following command:

```
Import-AzAutomationDscConfiguration  
  -ResourceGroupName $resourceGroup  
  -AutomationAccountName $automationAccount  
  -SourcePath "path\LinuxConfig.ps1"  
  -Published
```

The output should look similar as shown below:

```
ResourceGroupName : ContosoLab  
AutomationAccountName : MyAutomationAccount  
Location : westus  
State : Published  
Name : LinuxConfig  
Tags : {}  
CreationTime : 6/21/2021 10:02:11 PM +00:00  
LastModifiedTime : 6/21/2021 10:02:11 PM +00:00  
Description :  
Parameters : {}  
LogVerbose : False
```

You can view the configuration from your Automation account running the following command:

```
Get-AzAutomationDscConfiguration  
  -ResourceGroupName $resourceGroup  
  -AutomationAccountName $automationAccount  
  -Name $configurationName
```

Compile configuration in Azure Automation

Before you can apply a desired state to a node, the configuration defining that state must be compiled into one or more node configurations. Call the [Start-AzAutomationDscCompilationJob](#) cmdlet to compile the `LinuxConfig` configuration in Azure Automation. For more information about compilation, see [Compile DSC configurations](#). Run the following command:

```
Start-AzAutomationDscCompilationJob  
  -ResourceGroupName $resourceGroup  
  -AutomationAccountName $automationAccount  
  -ConfigurationName $configurationName
```

The output should look similar as shown below:

```
ResourceGroupName : ContosoLab  
AutomationAccountName : MyAutomationAccount  
Id : 11218f3c-f246-4fe9-bf5b-2123b9cad2e2  
CreationTime : 6/21/2021 10:07:13 PM +00:00  
Status : New  
StatusDetails : None  
StartTime :  
EndTime :  
Exception :  
LastModifiedTime : 6/21/2021 10:07:13 PM +00:00  
LastStatusModifiedTime : 6/21/2021 10:07:13 PM +00:00  
JobParameters : {}  
ConfigurationName : LinuxConfig
```

You can view the compilation job from your Automation account using the following command:

```
Get-AzAutomationDscCompilationJob ` 
    -ResourceGroupName $resourceGroup ` 
    -AutomationAccountName $automationAccount ` 
    -ConfigurationName $configurationName
```

Wait for the compilation job to complete before proceeding. The configuration must be compiled into a node configuration before it can be assigned to a node. Execute the following code to check for status every 5 seconds:

```
while ((Get-AzAutomationDscCompilationJob ` 
    -ResourceGroupName $resourceGroup ` 
    -AutomationAccountName $automationAccount ` 
    -ConfigurationName $configurationName).Status -ne "Completed") 
{ 
    Write-Output "Wait" 
    Start-Sleep -Seconds 5 
} 
Write-Output "Compilation complete"
```

After the compilation job completes, you can also view the node configuration metadata using the following command:

```
Get-AzAutomationDscNodeConfiguration ` 
    -ResourceGroupName $resourceGroup ` 
    -AutomationAccountName $automationAccount
```

Register the Azure Linux VM for an Automation account

Register the Azure Linux VM as a Desired State Configuration (DSC) node for the Azure Automation account. The [Register-AzAutomationDscNode](#) cmdlet only supports VMs running Windows OS. The Azure Linux VM will first need to be configured for DSC. For detailed steps, see [Get started with Desired State Configuration \(DSC\) for Linux](#).

1. Construct a python script with the registration command using PowerShell for later execution on your Azure Linux VM by running the following code:

```
$primaryKey = (Get-AzAutomationRegistrationInfo ` 
    -ResourceGroupName $resourceGroup ` 
    -AutomationAccountName $automationAccount).PrimaryKey

$URL = (Get-AzAutomationRegistrationInfo ` 
    -ResourceGroupName $resourceGroup ` 
    -AutomationAccountName $automationAccount).Endpoint

Write-Output "sudo /opt/microsoft/dsc/Scripts/Register.py $primaryKey $URL"
```

These commands obtain the Automation account's primary access key and URL and concatenates it to the registration command. Ensure you remove any carriage returns from the output. This command will be used in a later step.

2. Connect to your Azure Linux VM. If you used a password, you can use the syntax below. If you used a public-private key pair, see [SSH on Linux](#) for detailed steps. The other commands retrieve information about what packages can be installed, including what updates to currently installed packages packages are available, and installs Python.

```
ssh user@IP

sudo apt-get update
sudo apt-get install -y python
```

3. Install Open Management Infrastructure (OMI). For more information on OMI, see [Open Management Infrastructure](#). Verify the latest [release](#). Revise the release version below as needed, and then execute the commands in your ssh session:

```
wget https://github.com/microsoft/omi/releases/download/v1.6.8-0/omi-1.6.8-0.ssl_110.ulinux.x64.deb

sudo dpkg -i ./omi-1.6.8-0.ssl_110.ulinux.x64.deb
```

4. Install PowerShell Desired State Configuration for Linux. For more information, see [DSC on Linux](#). Verify the latest [release](#). Revise the release version below as needed, and then execute the commands in your ssh session:

```
wget https://github.com/microsoft/PowerShell-DSC-for-Linux/releases/download/v1.2.1-0/dsc-1.2.1-
0.ssl_110.x64.deb

sudo dpkg -i ./dsc-1.2.1-0.ssl_110.x64.deb
```

5. Now you can register the node using the

```
sudo /opt/microsoft/dsc/Scripts/Register.py <Primary Access Key> <URL>
```

Python script created in step 1. Run the commands in your ssh session, and the following output should look similar as shown below:

```
instance of SendConfigurationApply
{
    ReturnValue=0
}
```

6. You can verify the registration in PowerShell using the following command:

```
Get-AzAutomationDscNode `

-ResourceGroupName $resourceGroup `

-AutomationAccountName $automationAccount `

-Name $VM
```

The output should look similar as shown below:

```
ResourceGroupName      : ContosoLab
AutomationAccountName : MyAutomationAccount
Name                  : MyLinuxvM
RegistrationTime       : 6/21/2021 10:17:04 PM +00:00
LastSeen               : 6/21/2021 10:17:04 PM +00:00
IpAddress              : 127.0.0.1::1;172.19.0.9;fe80::20d:3aff:fe32:9082;
Id                    : 8c2e710a-421d-4405-8c01-9b32cbd56c73
NodeConfigurationName  :
Status                : Compliant
```

Assign a node configuration

Call the `Set-AzAutomationDscNode` cmdlet to set the node configuration mapping. Run the following commands:

```

# Get the ID of the DSC node
$node = Get-AzAutomationDscNode ` 
    -ResourceGroupName $resourceGroup ` 
    -AutomationAccountName $automationAccount ` 
    -Name $VM

# Set node configuration mapping
Set-AzAutomationDscNode ` 
    -ResourceGroupName $resourceGroup ` 
    -AutomationAccountName $automationAccount ` 
    -NodeConfigurationName $nodeConfigurationName0 ` 
    -NodeId $node.Id ` 
    -Force

```

The output should look similar as shown below:

```

ResourceGroupName      : ContosoLab
AutomationAccountName : MyAutomationAccount
Name                  : MyLinuxVM
RegistrationTime       : 6/21/2021 10:17:04 PM +00:00
LastSeen               : 6/21/2021 10:17:04 PM +00:00
IpAddress              : 127.0.0.1; ::1; 172.19.0.9; fe80::20d:3aff:fe32:9082;
Id                    : 8c2e710a-421d-4405-8c01-9b32cbd56c73
NodeConfigurationName  : LinuxConfig.IsNotPresent
Status                :

```

Modify the node configuration mapping

Call the [Set-AzAutomationDscNode](#) cmdlet to modify the node configuration mapping. Here, you modify the current node configuration mapping from `LinuxConfig.IsNotNull` to `LinuxConfig.isPresent`. Run the following command:

```

# Modify node configuration mapping
Set-AzAutomationDscNode ` 
    -ResourceGroupName $resourceGroup ` 
    -AutomationAccountName $automationAccount ` 
    -NodeConfigurationName $nodeConfigurationName1 ` 
    -NodeId $node.Id ` 
    -Force

```

Check the compliance status of a managed node

Each time State Configuration does a consistency check on a managed node, the node sends a status report back to the pull server. The following example uses the [Get-AzAutomationDscNodeReport](#) cmdlet to report on the compliance status of a managed node.

```

Get-AzAutomationDscNodeReport ` 
    -ResourceGroupName $resourceGroup ` 
    -AutomationAccountName $automationAccount ` 
    -NodeId $node.Id ` 
    -Latest

```

The output should look similar as shown below:

```
ResourceGroupName : ContosoLab
AutomationAccountName : MyAutomationAccount
StartTime : 6/21/2021 10:30:01 PM +00:00
LastModifiedTime : 6/21/2021 10:30:11 PM +00:00
EndTime : 6/21/2021 10:30:07 PM +00:00
ReportType : Consistency
Id : ceb3166b-4ad9-4d94-b00b-3b32b580ed87
NodeId : 8c2e710a-421d-4405-8c01-9b32cbd56c73
Status : Compliant
RefreshMode :
RebootRequested :
ReportFormatVersion : 2.0
```

The first report may not be available immediately and may take up to 30 minutes after you enable a node. For more information about report data, see [Using a DSC report server](#).

Clean up resources

The following steps help you delete the resources created for this tutorial that are no longer needed.

1. Remove DSC node from management by an Automation account. Although you can't register a node through PowerShell, you can unregister it with PowerShell. Run the following commands:

```
# Get the ID of the DSC node
$NodeID = (Get-AzAutomationDscNode ` 
    -ResourceGroupName $resourceGroup ` 
    -AutomationAccountName $automationAccount ` 
    -Name $VM).Id

Unregister-AzAutomationDscNode ` 
    -ResourceGroupName $resourceGroup ` 
    -AutomationAccountName $automationAccount ` 
    -Id $NodeID ` 
    -Force

# Verify using the same command from Register the Azure Linux VM for an Automation account. A blank
# response indicates success.
Get-AzAutomationDscNode ` 
    -ResourceGroupName $resourceGroup ` 
    -AutomationAccountName $automationAccount ` 
    -Name $VM
```

2. Remove metadata from DSC node configurations in Automation. Run the following commands:

```

Remove-AzAutomationDscNodeConfiguration ` 
-ResourceGroupName $resourceGroup ` 
-AutomationAccountName $automationAccount ` 
-Name $nodeConfigurationName0 ` 
-IgnoreNodeMappings ` 
-Force

Remove-AzAutomationDscNodeConfiguration ` 
-ResourceGroupName $resourceGroup ` 
-AutomationAccountName $automationAccount ` 
-Name $nodeConfigurationName1 ` 
-IgnoreNodeMappings ` 
-Force

# Verify using the same command from Compile configuration in Azure Automation.
Get-AzAutomationDscNodeConfiguration ` 
-ResourceGroupName $resourceGroup ` 
-AutomationAccountName $automationAccount ` 
-Name $nodeConfigurationName0

Get-AzAutomationDscNodeConfiguration ` 
-ResourceGroupName $resourceGroup ` 
-AutomationAccountName $automationAccount ` 
-Name $nodeConfigurationName1

```

Successful removal is indicated by output that looks similar to the following:

```
Get-AzAutomationDscNodeConfiguration : NodeConfiguration LinuxConfig.IsNotNullPresent not found .
```

3. Remove DSC configuration from Automation. Run the following command:

```

Remove-AzAutomationDscConfiguration ` 
-AutomationAccountName $automationAccount ` 
-ResourceGroupName $resourceGroup ` 
-Name $configurationName ` 
-Force

# Verify using the same command from Import configuration to Azure Automation.
Get-AzAutomationDscConfiguration ` 
-ResourceGroupName $resourceGroup ` 
-AutomationAccountName $automationAccount ` 
-Name $configurationName

```

Successful removal is indicated by output that looks similar to the following:

```
Get-AzAutomationDscConfiguration : Operation returned an invalid status code 'NotFound' .
```

4. Removes nx module from Automation. Run the following command:

```

Remove-AzAutomationModule ` 
-ResourceGroupName $resourceGroup ` 
-AutomationAccountName $automationAccount ` 
-Name $moduleName -Force

# Verify using the same command from Install nx module.
Get-AzAutomationModule ` 
-ResourceGroupName $resourceGroup ` 
-AutomationAccountName $automationAccount ` 
-Name $moduleName

```

Successful removal is indicated by output that looks similar to the following:

```
Get-AzAutomationModule : The module was not found. Module name: nx. .
```

Next steps

In this tutorial, you applied an Azure Automation State Configuration with PowerShell to an Azure Linux VM to check whether it complied with a desired state. For a more thorough explanation of configuration composition, see:

[Compose DSC configurations](#)

Get started with Azure Automation State Configuration

5/11/2021 • 8 minutes to read • [Edit Online](#)

This article provides a step-by-step guide for doing the most common tasks with Azure Automation State Configuration, such as creating, importing, and compiling configurations, enabling machines to manage, and viewing reports. For an overview State Configuration, see [State Configuration overview](#). For Desired State Configuration (DSC) documentation, see [Windows PowerShell Desired State Configuration Overview](#).

If you want a sample environment that is already set up without following the steps described in this article, you can use the [Azure Automation Managed Node template](#). This template sets up a complete State Configuration (DSC) environment, including an Azure VM that is managed by State Configuration (DSC).

Prerequisites

To complete the examples in this article, the following are required:

- An Azure Automation account. To learn more about an Automation account and its requirements, see [Automation Account authentication overview](#).
- An Azure Resource Manager VM (not Classic) running a [supported operating system](#). For instructions on creating a VM, see [Create your first Windows virtual machine in the Azure portal](#)

Create a DSC configuration

You create a simple [DSC configuration](#) that ensures either the presence or absence of the **Web-Server** Windows Feature (IIS), depending on how you assign nodes.

1. Start [VSCode](#) (or any text editor).
2. Type the following text:

```
configuration TestConfig
{
    Node IsWebServer
    {
        WindowsFeature IIS
        {
            Ensure          = 'Present'
            Name           = 'Web-Server'
            IncludeAllSubFeature = $true
        }
    }

    Node NotWebServer
    {
        WindowsFeature IIS
        {
            Ensure          = 'Absent'
            Name           = 'Web-Server'
        }
    }
}
```

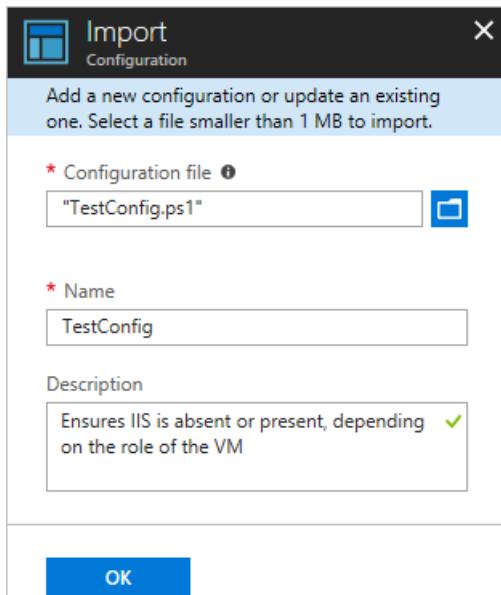
3. Save the file as **TestConfig.ps1**.

This configuration calls one resource in each node block, the [WindowsFeature](#) resource. This resource ensures either the presence or absence of the **Web-Server** feature.

Import a configuration into Azure Automation

Next, you import the configuration into the Automation account.

1. Sign in to the [Azure portal](#).
2. On the left, click **All resources** and then the name of your Automation account.
3. On the Automation account page, select **State configuration (DSC)** under **Configuration Management**.
4. On the State configuration (DSC) page, click the **Configurations** tab, then click **Add**.
5. On the Import Configuration pane, browse to the `TestConfig.ps1` file on your computer.

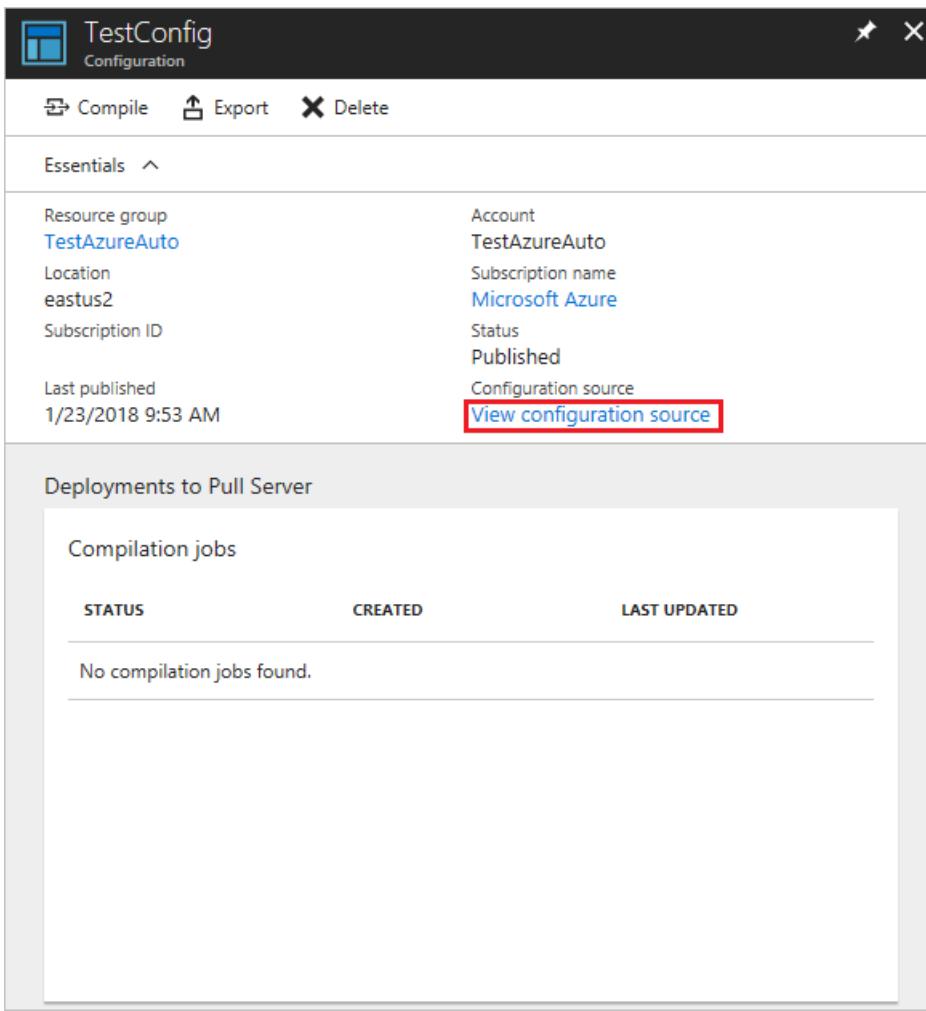


6. Click **OK**.

View a configuration in Azure Automation

After you have imported a configuration, you can view it in the Azure portal.

1. Sign in to the [Azure portal](#).
2. On the left, click **All resources** and then the name of your Automation account.
3. On the Automation account page, select **State configuration (DSC)** under **Configuration Management**.
4. On the State configuration (DSC) page, click the **Configurations** tab, then click **TestConfig**. This is the name of the configuration you imported in the previous procedure.
5. On the **TestConfig** Configuration pane, click **View configuration source**.



A TestConfig Configuration source pane opens, displaying the PowerShell code for the configuration.

Compile a configuration in Azure Automation

Before you can apply a desired state to a node, a DSC configuration defining that state must be compiled into one or more node configurations (MOF document), and placed on the Automation DSC Pull Server. For a more detailed description of compiling configurations in State Configuration (DSC), see [Compile configurations in Azure Automation State Configuration](#). For more information about compiling configurations, see [DSC Configurations](#).

1. Sign in to the [Azure portal](#).
2. On the left, click **All resources** and then the name of your Automation account.
3. On the Automation account page, click **State configuration (DSC)** under **Configuration Management**.
4. On the State configuration (DSC) page, click the **Configurations** tab, then click **TestConfig**. This is the name of the previously imported configuration.
5. On the TestConfig Configuration pane, click **Compile**, and then click **Yes**. This starts a compilation job.

The screenshot shows the Azure Automation Configuration pane for a configuration named 'TestConfig'. The top navigation bar includes 'Compile' (highlighted in red), 'Export', and 'Delete'. Below the navigation is a section titled 'Essentials' with the following details:

Resource group	Account
TestAzureAuto	TestAzureAuto
Location	Subscription name
eastus2	Microsoft Azure
Subscription ID	Status
	Published
Last published	Configuration source
1/23/2018 9:53 AM	View configuration source

Below this is a section titled 'Deployments to Pull Server' containing a table for 'Compilation jobs':

STATUS	CREATED	LAST UPDATED
No compilation jobs found.		

NOTE

When you compile a configuration in Azure Automation, it automatically deploys any created node configuration MOF files to the pull server.

View a compilation job

After you start a compilation, you can view it in the **Compilation Jobs** tile on the **Configuration** page. The **Compilation Jobs** tile shows currently running, completed, and failed jobs. When you open a compilation job pane, it shows information about that job including any errors or warnings encountered, input parameters used in the configuration, and compilation logs.

1. Sign in to the [Azure portal](#).
2. On the left, click **All resources** and then the name of your Automation account.
3. On the Automation account page, click **State configuration (DSC)** under **Configuration Management**.
4. On the State configuration (DSC) page, click the **Configurations** tab, then click **TestConfig**. This is the name of the previously imported configuration.
5. Under **Compilation jobs**, select the compilation job to view. A Compilation Job pane opens, labeled with the date when the compilation job was started.

1/23/2018 10:08 AM
Compilation Job

Essentials

Resource group	Account
TestAzureAuto	TestAzureAuto
Location	Subscription name
eastus2	Microsoft Azure
State	Start time
Completed	1/23/2018 10:10 AM
Last Update	Total runtime
1/23/2018 10:10 AM	Less than one minute
Job Id	Configuration
83d2f2f3-f87e-4f0d-b10d-973150d44b24	TestConfig

Details

Input

0 ➔ Configuration source snapshot

</>

Monitoring

Errors: 0 ✗ Warnings: 2 ⚠ All Logs

- Click on any tile in the Compilation Job pane to see further details about the job.

View node configurations

Successful completion of a compilation job creates one or more new node configurations. A node configuration is a MOF document that is deployed to the pull server and ready to be pulled and applied by one or more nodes. You can view the node configurations in your Automation account on the State configuration (DSC) page. A node configuration has a name with the form `ConfigurationName.NodeName`.

- Sign in to the [Azure portal](#).
- On the left, click **All resources** and then the name of your Automation account.
- On the Automation account page, click **State configuration (DSC)** under **Configuration Management**.
- On the State configuration (DSC) page, click the **Compiled configurations** tab.

Home > automation - State configuration (DSC)

automation - State configuration (DSC)

Automation Account

State Config

CONFIGURATION MANAGEMENT

State configuration (DSC)

+ Add ⌛ Refresh ⌂ Reset filters

Nodes Configurations Compiled configurations Gallery

Node configuration ⓘ

Search node configurations...

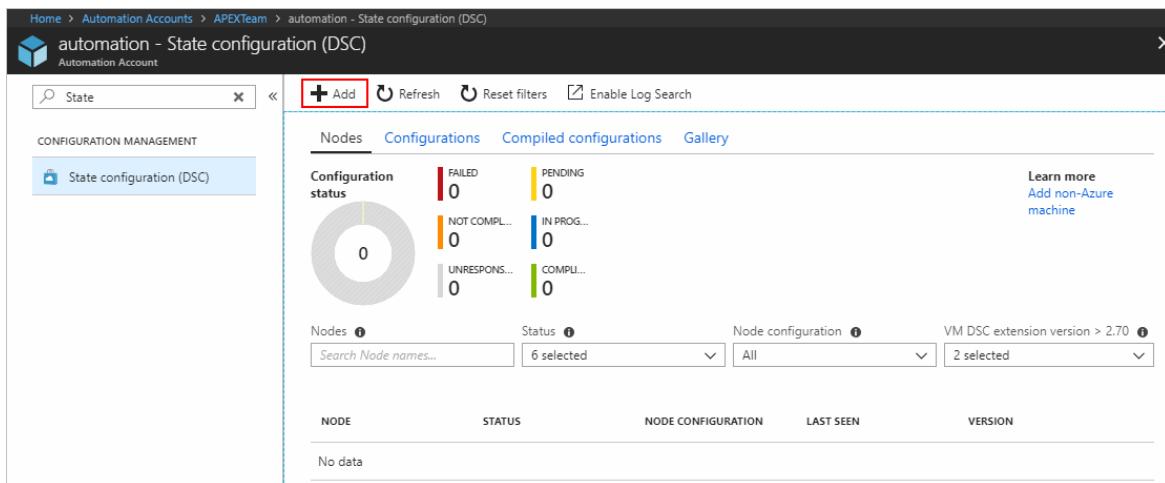
NODE CONFIGURATION	CONFIGURATION	NODE COUNT	CREATED	LAST MODIFIED
TestConfig.IISWebServer	TestConfig	0	7/29/2018, 1:43 PM	7/29/2018, 1:43 PM ...
TestConfig.NotWebServer	TestConfig	0	7/29/2018, 1:43 PM	7/29/2018, 1:43 PM ...

Enable an Azure Resource Manager VM for management with State

Configuration

You can use State Configuration to manage Azure VMs (both classic and Resource Manager), on-premises VMs, Linux machines, AWS VMs, and on-premises physical machines. In this article, you learn how to enable only Azure Resource Manager VMs. For information about enabling other types of machines, see [Enable machines for management by Azure Automation State Configuration](#).

1. Sign in to the [Azure portal](#).
2. On the left, click **All resources** and then the name of your Automation account.
3. On the Automation account page, click **State configuration (DSC)** under **Configuration Management**.
4. On the State configuration (DSC) page, select the **Nodes** tab, then click **+ Add**.



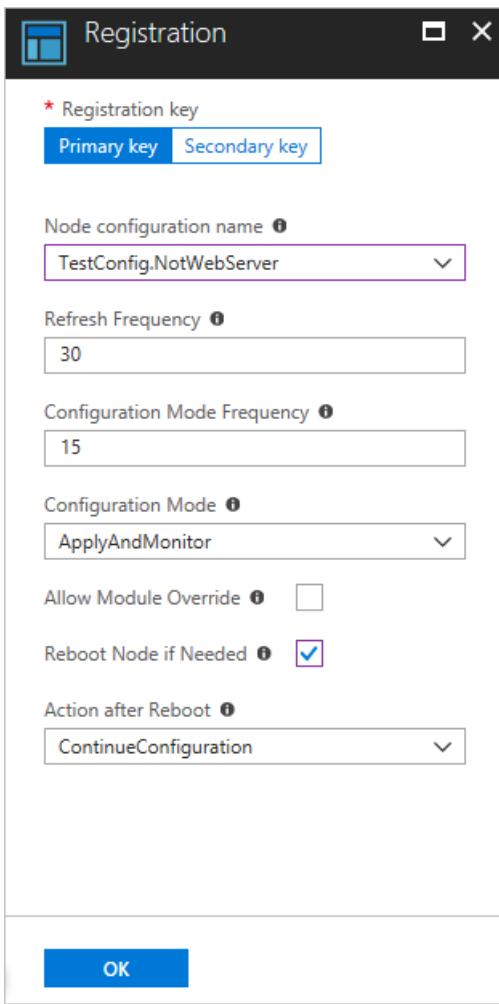
The screenshot shows the 'Nodes' tab in the State configuration (DSC) section of the Azure portal. At the top, there's a navigation bar with 'Home > Automation Accounts > APEXTeam > automation - State configuration (DSC)'. Below it is a header with 'automation - State configuration (DSC)' and 'Automation Account'. A search bar says 'State'. To the right of the search bar is a red box around the '+ Add' button. There are also 'Refresh', 'Reset filters', and 'Enable Log Search' buttons. The main area has tabs for 'Nodes', 'Configurations', 'Compiled configurations', and 'Gallery'. Under 'Nodes', there's a summary chart with a central circle containing '0' and four colored bars: red for 'FAILED' (0), yellow for 'PENDING' (0), orange for 'NOT COMPL...' (0), and green for 'UNRESPONS...' (0). To the right is a link 'Learn more Add non-Azure machine'. Below the chart are filters for 'Nodes', 'Status', 'Node configuration', and 'VM DSC extension version > 2.70'. At the bottom is a table with columns 'NODE', 'STATUS', 'NODE CONFIGURATION', 'LAST SEEN', and 'VERSION', showing 'No data'.

5. On the Virtual Machines pane, select your VM.
6. On the Virtual machine detail pane, click **+ Connect**.

IMPORTANT

The VM must be an Azure Resource Manager VM running a [supported operating system](#).

7. On the Registration page, select the name of the node configuration to apply to the VM in the **Node configuration name** field. Providing a name at this point is optional. You can change the assigned node configuration after enabling the node.
8. Check **Reboot Node if Needed**, then click **OK**.



The node configuration you specified is applied to the VM at intervals specified by the value provided for **Configuration Mode Frequency**. The VM checks for updates to the node configuration at intervals specified by the **Refresh Frequency** value. For more information about how these values are used, see [Configuring the Local Configuration Manager](#).

Azure starts the process of enabling the VM. When it is complete, the VM shows up in the **Nodes** tab of the State configuration (DSC) page in the Automation account.

View the list of managed nodes

You can view the list of all machines that have been enabled for management in your Automation account in the **Nodes** tab of the State configuration (DSC) page.

1. Sign in to the [Azure portal](#).
2. On the left, click **All resources** and then the name of your Automation account.
3. On the Automation account page, click **State configuration (DSC)** under **Configuration Management**.
4. On the State configuration (DSC) page, click the **Nodes** tab.

View reports for managed nodes

Each time State Configuration performs a consistency check on a managed node, the node sends a status report back to the pull server. You can view these reports on the page for that node.

1. Sign in to the [Azure portal](#).
2. On the left, click **All resources** and then the name of your Automation account.
3. On the Automation account page, click **State configuration (DSC)** under **Configuration**

Management.

4. On the State configuration (DSC) page, click the **Nodes** tab. Here, you can see the overview of Configuration state and the details for each node.

NODE	STATUS	NODE CONFIGURATION	LAST SEEN	VERSION
ContosoVM1	Compliant	TestConfig.NotWebServer	7/29/2018, 2:15 PM	2.76.0.0

5. While on the **Nodes** tab, click the node record to open the reporting. Click the report you want to view additional reporting details.

TYPE	STATUS	REPORT TIME
Consistency	✓ Compliant	12/7/2017 9:30 AM
Consistency	✓ Compliant	12/7/2017 9:14 AM
Consistency	✓ Compliant	12/7/2017 9:14 AM
Consistency	✓ Compliant	12/7/2017 8:59 AM
Consistency	✓ Compliant	12/7/2017 8:44 AM
Consistency	✓ Compliant	12/7/2017 8:44 AM

On the blade for an individual report, you can see the following status information for the corresponding consistency check:

- The report status. Possible values are:
 - Compliant - the node is compliant with the check.
 - Failed - the configuration failed the check.
 - Not Compliant - the node is in `ApplyandMonitor` mode and the machine is not in the desired state.
- The start time for the consistency check.
- The total runtime for the consistency check.
- The type of consistency check.
- Any errors, including the error code and error message.
- Any DSC resources used in the configuration, and the state of each resource (whether the node is in the desired state for that resource). You can click on each resource to get more detailed information for that

resource.

- The name, IP address, and configuration mode of the node.

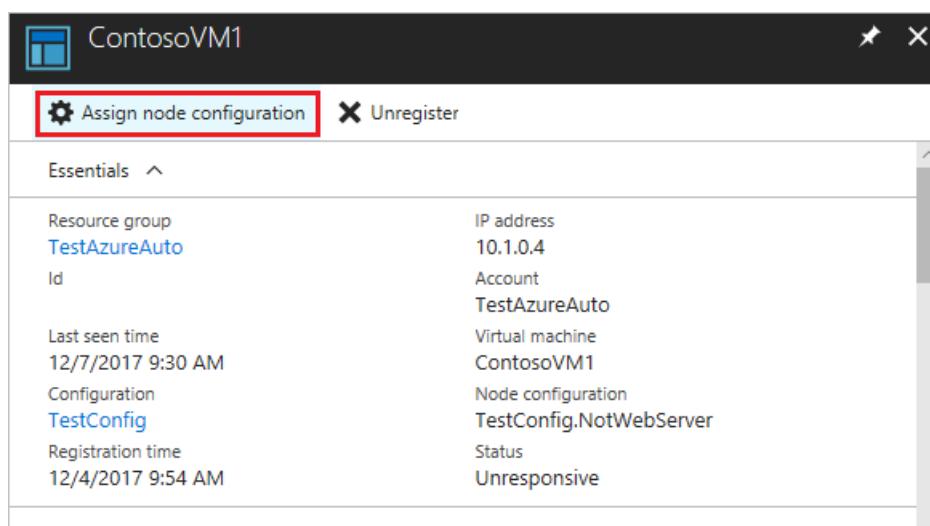
You can also click **View raw report** to see the actual data that the node sends to the server. For more information about using that data, see [Using a DSC report server](#).

It can take some time after a node is enabled before the first report is available. You might need to wait up to 30 minutes for the first report after you enable a node.

Reassign a node to a different node configuration

You can assign a node to use a different node configuration than the one you initially assigned.

1. Sign in to the [Azure portal](#).
2. On the left, click **All resources** and then the name of your Automation account.
3. On the Automation account page, click **State configuration (DSC)** under Configuration Management.
4. On the State configuration (DSC) page, click the **Nodes** tab.
5. On the **Nodes** tab, click on the name of the node you want to reassign.
6. On the page for that node, click **Assign node configuration**.



The screenshot shows the Azure portal interface for a node named 'ContosoVM1'. The title bar says 'ContosoVM1'. Below it, there are two buttons: 'Assign node configuration' (highlighted with a red box) and 'Unregister'. The main area is titled 'Essentials' and contains the following data:

Resource group	IP address
TestAzureAuto	10.1.0.4
Id	Account
	TestAzureAuto
Last seen time	Virtual machine
12/7/2017 9:30 AM	ContosoVM1
Configuration	Node configuration
TestConfig	TestConfig.NotWebServer
Registration time	Status
12/4/2017 9:54 AM	Unresponsive

7. On the Assign Node Configuration page, select the node configuration to which you want to assign the node, and then click **OK**.

 Assign Node Configuration
ContosoVM1

Changing the node configuration assigned to a node will cause the node to change its configuration to match the node configuration next time it pulls.

NAME	LAST MODIFIED
TestConfig.NotWebServer	1/23/2018 10:19 AM
TestConfig.IsWebServer	1/23/2018 10:21 AM

OK

Unregister a node

If you no longer want a node to be managed by State Configuration, you can unregister it. See [How to remove a configuration and node from Automation State Configuration](#).

Next steps

- For an overview, see [Azure Automation State Configuration overview](#).
- To enable the feature for VMs in your environment, see [Enable Azure Automation State Configuration](#).
- To understand PowerShell DSC, see [Windows PowerShell Desired State Configuration overview](#).
- For pricing information, see [Azure Automation State Configuration pricing](#).
- For a PowerShell cmdlet reference, see [Az.Automation](#).

Enable Azure Automation State Configuration

7/12/2021 • 12 minutes to read • [Edit Online](#)

This topic describes how you can set up your machines for management with Azure Automation State Configuration. For details of this service, see [Azure Automation State Configuration overview](#).

Enable Azure VMs

Azure Automation State Configuration lets you easily enable Azure VMs for configuration management, using the Azure portal, Azure Resource Manager templates, or PowerShell. Under the hood, and without an administrator having to remote into a VM, the Azure VM Desired State Configuration extension registers the VM with Azure Automation State Configuration. Since the Azure extension runs asynchronously, steps to track its progress are provided in [Check status of VM setup](#).

NOTE

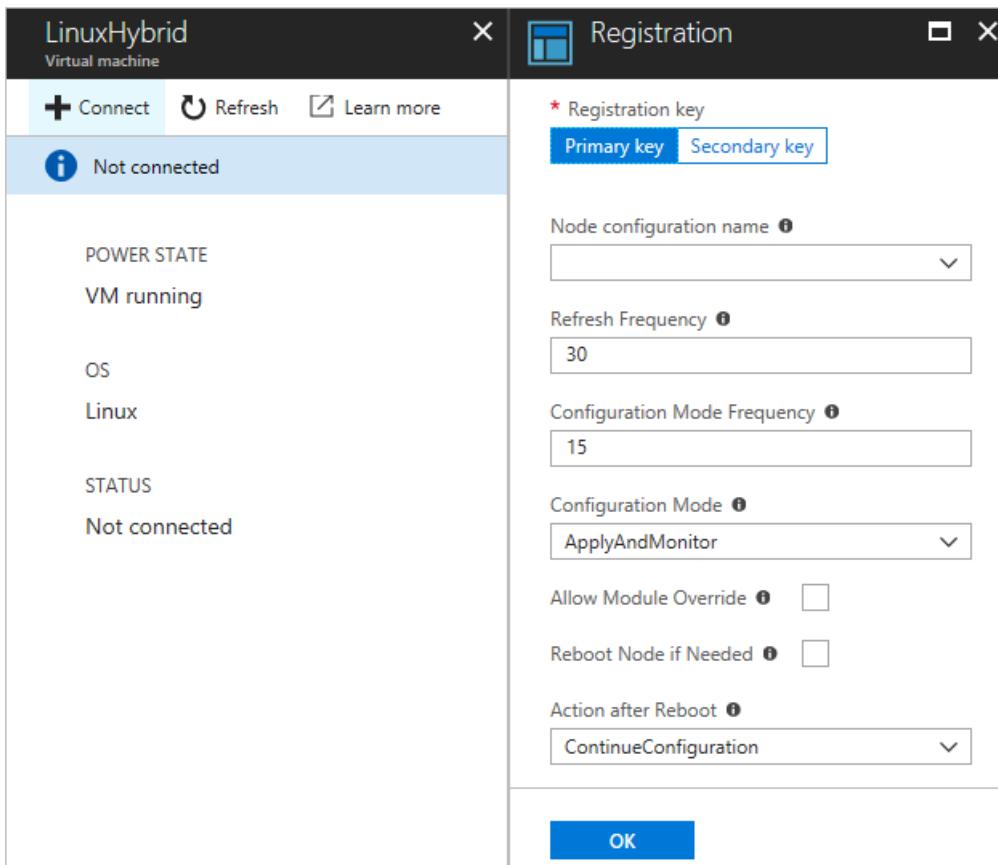
Deploying DSC to a Linux node uses the `/tmp` folder. Modules such as `nxautomation` are temporarily downloaded for verification before installing them in their appropriate locations. To ensure that modules install correctly, the Log Analytics agent for Linux needs read/write permissions on the `/tmp` folder.

The Log Analytics agent for Linux runs as the `omsagent` user. To grant >write permission to the `omsagent` user, run the command `setfacl -m u:omsagent:rwx /tmp`.

Enable a VM using Azure portal

To enable an Azure VM to State Configuration through the [Azure portal](#):

1. Navigate to the Azure Automation account in which to enable VMs.
2. On the State Configuration page, select the **Nodes** tab, then click **Add**.
3. Choose a VM to enable.
4. If the machine doesn't have the PowerShell desired state extension installed and the power state is running, click **Connect**.
5. Under **Registration**, enter the [PowerShell DSC Local Configuration Manager values](#) required for your use case. Optionally, you can enter a node configuration to assign to the VM.



Enable a VM using Azure Resource Manager templates

You can install and enable a VM for State Configuration using Azure Resource Manager templates. See [Server managed by Desired State Configuration service](#) for an example template that enables an existing VM for State Configuration. If you are managing a virtual machine scale set, see the example template in [Virtual machine scale set configuration managed by Azure Automation](#).

Enable machines using PowerShell

You can use the `Register-AzAutomationDscNode` cmdlet in PowerShell to enable VMs for State Configuration.

NOTE

The `Register-AzAutomationDscNode` cmdlet is implemented currently only for machines running Windows, as it triggers just the Windows extension.

Register VMs across Azure subscriptions

The best way to register VMs from other Azure subscriptions is to use the DSC extension in an Azure Resource Manager deployment template. Examples are provided in [Desired State Configuration extension with Azure Resource Manager templates](#).

To find the registration key and registration URL to use as parameters in the template, see [Enable machines securely using registration](#).

Enable physical/virtual Windows machines

You can enable Windows servers running on-premises or in other cloud environments (including AWS EC2 instances) to Azure Automation State Configuration. The servers must have [outbound access to Azure](#).

1. Make sure that the latest version of [WMF 5](#) is installed on the machines to enable for State Configuration. In addition, WMF 5 must be installed on the computer that you are using for enabling the machines.
2. Follow the directions in [Generate DSC metaconfigurations](#) to create a folder containing the required DSC

metaconfigurations.

3. Use the following cmdlet to apply the PowerShell DSC metaconfigurations remotely to the machines to enable.

```
Set-DscLocalConfigurationManager -Path C:\Users\joe\Desktop\DscMetaConfigs -ComputerName MyServer1, MyServer2
```

4. If you can't apply the PowerShell DSC metaconfigurations remotely, copy the **metaconfigurations** folder to the machines that you are enabling. Then add code to call [Set-DscLocalConfigurationManager](#) locally on the machines.
5. Using the Azure portal or cmdlets, verify that the machines appear as State Configuration nodes registered in your Azure Automation account.

Enable physical/virtual Linux machines

You can enable Linux servers running on-premises or in other cloud environments for State Configuration. The servers must have [outbound access to Azure](#).

1. Make sure that the latest version of [PowerShell Desired State Configuration for Linux](#) is installed on the machines to enable for State Configuration.
2. If the [PowerShell DSC Local Configuration Manager defaults](#) match your use case, and you want to enable machines so that they both pull from and report to State Configuration:
 - On each Linux machine to enable, use `Register.py` to enable the machine with the PowerShell DSC Local Configuration Manager defaults.

```
/opt/microsoft/dsc/Scripts/Register.py <Automation account registration key> <Automation account registration URL>
```
 - To find the registration key and registration URL for your Automation account, see [Enable machines securely using registration](#).
3. If the PowerShell DSC Local Configuration Manager (LCM) defaults don't match your use case, or you want to enable machines that only report to Azure Automation State Configuration, follow steps 4-7. Otherwise, proceed directly to step 7.
4. Follow the directions in [Generate DSC metaconfigurations](#) section to produce a folder containing the required DSC metaconfigurations.
5. Make sure that the latest version of [WMF 5](#) is installed on the computer being used to enable your machines for State Configuration.
6. Add code as follows to apply the PowerShell DSC metaconfigurations remotely to the machines to enable.

```
$SecurePass = ConvertTo-SecureString -String '<root password>' -AsPlainText -Force
$Cred = New-Object System.Management.Automation.PSCredential 'root', $SecurePass
$Opt = New-CimSessionOption -UseSsl -SkipCACheck -SkipCNCheck -SkipRevocationCheck

# need a CimSession for each Linux machine to onboard
$Session = New-CimSession -Credential $Cred -ComputerName <your Linux machine> -Port 5986 -
Authentication basic -SessionOption $Opt

Set-DscLocalConfigurationManager -CimSession $Session -Path C:\Users\joe\Desktop\DscMetaConfigs
```

7. If you can't apply the PowerShell DSC metaconfigurations remotely, copy the metaconfigurations

corresponding to the remote machines from the folder described in step 4 to the Linux machines.

8. Add code to call `Set-DscLocalConfigurationManager.py` locally on each Linux machine to enable for State Configuration.

```
/opt/microsoft/dsc/Scripts/SetDscLocalConfigurationManager.py -configurationmof <path to metaconfiguration file>
```

9. Using the Azure portal or cmdlets, ensure that the machines to enable now show up as DSC nodes registered in your Azure Automation account.

Generate DSC metaconfigurations

To enable any machine for State Configuration, you can generate a [DSC metaconfiguration](#). This configuration tells the DSC agent to pull from and/or report to Azure Automation State Configuration. You can generate a DSC metaconfiguration for Azure Automation State Configuration using either a PowerShell DSC configuration or the Azure Automation PowerShell cmdlets.

NOTE

DSC metaconfigurations contain the secrets needed to enable a machine in an Automation account for management. Make sure to properly protect any DSC metaconfigurations you create, or delete them after use.

Proxy support for metaconfigurations is controlled by the [Local Configuration Manager](#), which is the Windows PowerShell DSC engine. The LCM runs on all target nodes and is responsible for calling the configuration resources that are included in a DSC metaconfiguration script. You can include proxy support in a metaconfiguration by including definitions of `ProxyURL` and `ProxyCredential` properties as needed in the `ConfigurationRepositoryWeb`, `ResourceRepositoryWeb`, and `ReportServerWeb` blocks. An example of the URL setting is `ProxyURL = "http://172.16.3.6:3128";`. The `ProxyCredential` property is set to a `PSCredential` object, as described in [Manage credentials in Azure Automation](#).

Generate DSC metaconfigurations using a DSC configuration

1. Open VSCode (or your favorite editor) as an administrator on a machine in your local environment. The machine must have the latest version of [WMF 5](#) installed.
2. Copy the following script locally. This script contains a PowerShell DSC configuration for creating metaconfigurations, and a command to kick off the metaconfiguration creation.

NOTE

State Configuration Node Configuration names are case-sensitive in the Azure portal. If the case is mismatched, the node will not show up under the **Nodes** tab.

```
# The DSC configuration that will generate metaconfigurations
[DscLocalConfigurationManager()]
Configuration DscMetaConfigs
{
    param
    (
        [Parameter(Mandatory=$True)]
        [String]$RegistrationUrl,
        [Parameter(Mandatory=$True)]
        [String]$RegistrationKey,
        [Parameter(Mandatory=$True)]
        [String[]]$ComputerName,
```

```

[Int]$RefreshFrequencyMins = 30,
[Int]$ConfigurationModeFrequencyMins = 15,
[String]$ConfigurationMode = 'ApplyAndMonitor',
[String]$NodeConfigurationName,
[Boolean]$RebootNodeIfNeeded= $False,
[String]$ActionAfterReboot = 'ContinueConfiguration',
[Boolean]$AllowModuleOverwrite = $False,
[Boolean]$ReportOnly
)

if(!$NodeConfigurationName -or $NodeConfigurationName -eq '')
{
    $ConfigurationNames = $null
}
else
{
    $ConfigurationNames = @($NodeConfigurationName)
}

if($ReportOnly)
{
    $RefreshMode = 'PUSH'
}
else
{
    $RefreshMode = 'PULL'
}

Node $ComputerName
{
    Settings
    {
        RefreshFrequencyMins      = $RefreshFrequencyMins
        RefreshMode                = $RefreshMode
        ConfigurationMode          = $ConfigurationMode
        AllowModuleOverwrite       = $AllowModuleOverwrite
        RebootNodeIfNeeded          = $RebootNodeIfNeeded
        ActionAfterReboot           = $ActionAfterReboot
        ConfigurationModeFrequencyMins = $ConfigurationModeFrequencyMins
    }

    if(!$ReportOnly)
    {
        ConfigurationRepositoryWeb AzureAutomationStateConfiguration
        {
            ServerUrl      = $RegistrationUrl
            RegistrationKey = $RegistrationKey
            ConfigurationNames = $ConfigurationNames
        }

        ResourceRepositoryWeb AzureAutomationStateConfiguration
        {
            ServerUrl      = $RegistrationUrl
            RegistrationKey = $RegistrationKey
        }
    }

    ReportServerWeb AzureAutomationStateConfiguration
    {
        ServerUrl      = $RegistrationUrl
        RegistrationKey = $RegistrationKey
    }
}

```

```

    }

}

# Create the metaconfigurations
# NOTE: DSC Node Configuration names are case sensitive in the portal.
# TODO: edit the below as needed for your use case
$Params = @{
    RegistrationUrl = '<fill me in>';
    RegistrationKey = '<fill me in>';
    ComputerName = @('<some VM to onboard>', '<some other VM to onboard>');
    NodeConfigurationName = 'SimpleConfig.webserver';
    RefreshFrequencyMins = 30;
    ConfigurationModeFrequencyMins = 15;
    RebootNodeIfNeeded = $False;
    AllowModuleOverwrite = $False;
    ConfigurationMode = 'ApplyAndMonitor';
    ActionAfterReboot = 'ContinueConfiguration';
    ReportOnly = $False; # Set to $True to have machines only report to AA DSC but not pull from it
}

# Use PowerShell splatting to pass parameters to the DSC configuration being invoked
# For more info about splatting, run: Get-Help -Name about_Splatting
DscMetaConfigs @Params

```

3. Fill in the registration key and URL for your Automation account, as well as the names of the machines to enable. All other parameters are optional. To find the registration key and registration URL for your Automation account, see [Enable machines securely using registration](#).
4. If you want the machines to report DSC status information to Azure Automation State Configuration, but not pull configuration or PowerShell modules, set the `ReportOnly` parameter to true.
5. If `ReportOnly` is not set, the machines report DSC status information to Azure Automation State Configuration and pull configuration or PowerShell modules. Set parameters accordingly in the `ConfigurationRepositoryWeb`, `ResourceRepositoryWeb`, and `ReportServerWeb` blocks.
6. Run the script. You should now have a working directory folder called `DscMetaConfigs`, containing the PowerShell DSC metaconfigurations for the machines to enable (as an administrator).

```
Set-DscLocalConfigurationManager -Path ./DscMetaConfigs
```

Generate DSC metaconfigurations using Azure Automation cmdlets

If PowerShell DSC LCM defaults match your use case and you want to enable machines to both pull from and report to Azure Automation State Configuration, you can generate the needed DSC metaconfigurations more simply using the Azure Automation cmdlets.

1. Open the PowerShell console or VSCode as an administrator on a machine in your local environment.
2. Connect to Azure Resource Manager using [Connect-AzAccount](#).
3. Download the PowerShell DSC metaconfigurations for the machines you want to enable from the Automation account in which you are setting up nodes.

```
# Define the parameters for Get-AzAutomationDscOnboardingMetaconfig using PowerShell Splatting
$Params = @{
    ResourceGroupName = 'ContosoResources'; # The name of the Resource Group that contains your Azure Automation account
    AutomationAccountName = 'ContosoAutomation'; # The name of the Azure Automation account where you want a node on-boarded to
    ComputerName = @('web01', 'web02', 'sql01'); # The names of the computers that the metaconfiguration will be generated for
    OutputFolder = "$env:UserProfile\Desktop\";
}
# Use PowerShell splatting to pass parameters to the Azure Automation cmdlet being invoked
# For more info about splatting, run: Get-Help -Name about_Splatting
Get-AzAutomationDscOnboardingMetaconfig @Params
```

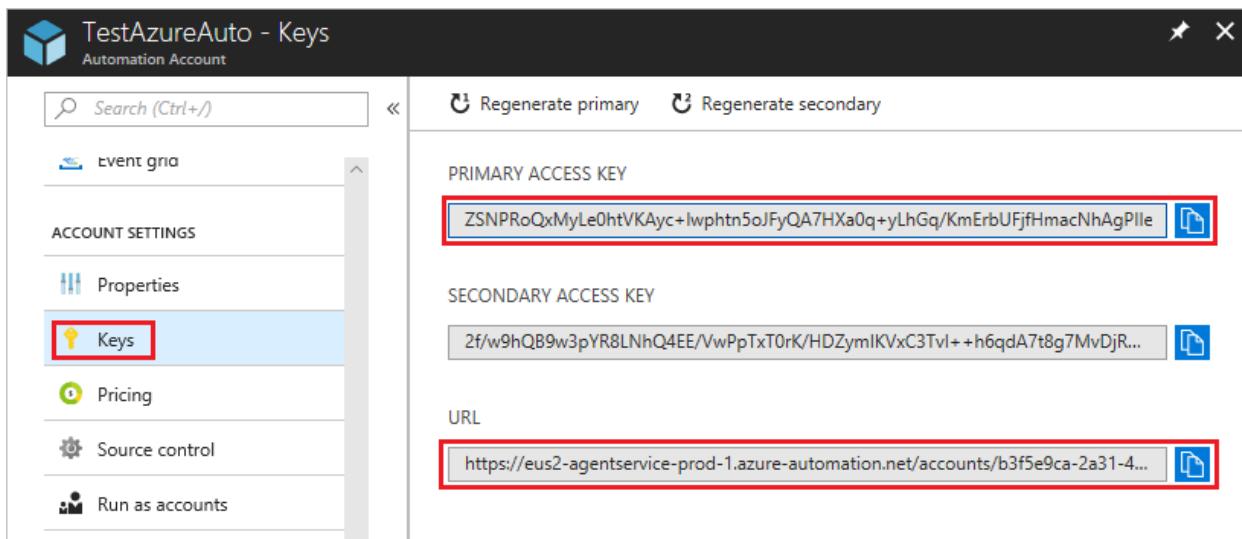
4. You should now have a **DscMetaConfigs** folder containing the PowerShell DSC metaconfigurations for the machines to enable (as an administrator).

```
Set-DscLocalConfigurationManager -Path $env:UserProfile\Desktop\DscMetaConfigs
```

Enable machines securely using registration

You can enable machines securely for an Azure Automation account through the WMF 5 DSC registration protocol. This protocol allows a DSC node to authenticate to a PowerShell DSC pull or report server, including Azure Automation State Configuration. The node registers with the server at the registration URL and authenticates using a registration key. During registration, the DSC node and DSC pull/report server negotiate a unique certificate for the node to use for authentication to the server post-registration. This process prevents enabled nodes from impersonating one another, for example, if a node is compromised and behaving maliciously. After registration, the registration key is not used for authentication again, and is deleted from the node.

You can get the information required for the State Configuration registration protocol from **Keys** under **Account Settings** in the Azure portal.



- Registration URL is the URL field on the Keys page.
- Registration key is the value of the **Primary access key** field or the **Secondary access key** field on the Keys page. Either key can be used.

For added security, you can regenerate the primary and secondary access keys of an Automation account at any time on the Keys page. Key regeneration prevents future node registrations from using previous keys.

Re-register a node

After registering a machine as a DSC node in Azure Automation State Configuration, there are several reasons why you might need to re-register that node in the future.

- **Certificate renewal.** For versions of Windows Server before Windows Server 2019, each node automatically negotiates a unique certificate for authentication that expires after one year. If a certificate expires without renewal, the node is unable to communicate with Azure Automation and is marked `Unresponsive`. Currently, the PowerShell DSC registration protocol can't automatically renew certificates when they are nearing expiration, and you must re-register the nodes after a year's time. Before re-registering, ensure that each node is running WMF 5 RTM.

Re-registration performed 90 days or less from the certificate expiration time, or at any point after the certificate expiration time, results in a new certificate being generated and used. A resolution to this issue is included in Windows Server 2019 and later.

- **Changes to DSC LCM values.** You might need to change [PowerShell DSC LCM values](#) set during initial registration of the node, for example, `ConfigurationMode`. Currently, you can only change these DSC agent values through re-registration. The one exception is the Node Configuration value assigned to the node. You can change this in Azure Automation DSC directly.

You can re-register a node just as you registered the node initially, using any of the methods described in this document. You do not need to unregister a node from Azure Automation State Configuration before re-registering it.

Check status of VM setup

State Configuration lets you easily enable Azure Windows VMs for configuration management. Under the hood, the Azure VM Desired State Configuration extension is used to register the VM with Azure Automation State Configuration. Since the Azure VM Desired State Configuration extension runs asynchronously, tracking its progress and troubleshooting its execution can be important.

NOTE

Any method of enabling Azure Windows VMs for State Configuration that uses the Azure VM Desired State Configuration extension can take up to an hour for Azure Automation to show VMs as registered. This delay is due to the installation of WMF 5 on the VM by the Azure VM Desired State Configuration extension, which is required to enable VMs for State Configuration.

To view the status of the Azure VM Desired State Configuration extension:

1. In the Azure portal, navigate to the VM being enabled.
2. Click **Extensions** under **Settings**.
3. Now select **DSC** or **DSCForLinux**, depending on your operating system.
4. For more details, you can click **View detailed status**.

Next steps

- To get started, see [Get started with Azure Automation State Configuration](#).
- To learn about compiling DSC configurations so that you can assign them to target nodes, see [Compile DSC configurations in Azure Automation State Configuration](#).
- For a PowerShell cmdlet reference, see [Az.Automation](#).
- For pricing information, see [Azure Automation State Configuration pricing](#).
- For an example of using Azure Automation State Configuration in a continuous deployment pipeline, see [Set](#)

up continuous deployment with Chocolatey.

- For troubleshooting information, see [Troubleshoot Azure Automation State Configuration](#).

Configure machines to a desired state

4/26/2021 • 4 minutes to read • [Edit Online](#)

Azure Automation State Configuration allows you to specify configurations for your servers and ensure that those servers are in the specified state over time.

- Onboard a VM to be managed by Azure Automation DSC
- Upload a configuration to Azure Automation
- Compile a configuration into a node configuration
- Assign a node configuration to a managed node
- Check the compliance status of a managed node

For this tutorial, we use a simple [DSC configuration](#) that ensures that IIS is installed on the VM.

Prerequisites

- An Azure Automation account. To learn more about an Automation account and its requirements, see [Automation Account authentication overview](#).
- An Azure Resource Manager VM (not classic) running Windows Server 2008 R2 or later. For instructions on creating a VM, see [Create your first Windows virtual machine in the Azure portal](#).
- Azure PowerShell module version 3.6 or later. Run `Get-Module -ListAvailable Az` to find the version. If you need to upgrade, see [Install Azure PowerShell module](#).
- Familiarity with Desired State Configuration (DSC). For information about DSC, see [Windows PowerShell Desired State Configuration Overview](#).

Support for partial configurations

Azure Automation State Configuration supports the use of [partial configurations](#). In this scenario, DSC is configured to manage multiple configurations independently, and each configuration is retrieved from Azure Automation. However, only one configuration can be assigned to a node per automation account. This means if you are using two configurations for a node you will require two Automation accounts.

For details about how to register a partial configuration from a pull service, see the documentation for [partial configurations](#).

For more information about how teams can work together to collaboratively manage servers using configuration as code, see [Understanding DSC's role in a CI/CD Pipeline](#).

Log in to Azure

Log in to your Azure subscription with the `Connect-AzAccount` cmdlet and follow the on-screen directions.

```
Connect-AzAccount
```

Create and upload a configuration to Azure Automation

In a text editor, type the following and save it locally as `TestConfig.ps1`.

```
configuration TestConfig {
    Node WebServer {
        WindowsFeature IIS {
            Ensure          = 'Present'
            Name           = 'Web-Server'
            IncludeAllSubFeature = $true
        }
    }
}
```

NOTE

In more advanced scenarios where you require multiple modules to be imported that provide DSC Resources, make sure each module has a unique `Import-DscResource` line in your configuration.

Call the [Import-AzAutomationDscConfiguration](#) cmdlet to upload the configuration into your Automation account.

```
Import-AzAutomationDscConfiguration -SourcePath 'C:\DscConfigs\TestConfig.ps1' -ResourceGroupName 'MyResourceGroup' -AutomationAccountName 'myAutomationAccount' -Published
```

Compile a configuration into a node configuration

A DSC configuration must be compiled into a node configuration before it can be assigned to a node. See [DSC configurations](#).

Call the [Start-AzAutomationDscCompilationJob](#) cmdlet to compile the `TestConfig` configuration into a node configuration named `TestConfig.WebServer` in your Automation account.

```
Start-AzAutomationDscCompilationJob -ConfigurationName 'TestConfig' -ResourceGroupName 'MyResourceGroup' -AutomationAccountName 'myAutomationAccount'
```

Register a VM to be managed by State Configuration

You can use Azure Automation State Configuration to manage Azure VMs (both Classic and Resource Manager), on-premises VMs, Linux machines, AWS VMs, and on-premises physical machines. In this topic, we cover how to register only Azure Resource Manager VMs. For information about registering other types of machines, see [Onboarding machines for management by Azure Automation State Configuration](#).

Call the [Register-AzAutomationDscNode](#) cmdlet to register your VM with Azure Automation State Configuration as a managed node.

```
Register-AzAutomationDscNode -ResourceGroupName 'MyResourceGroup' -AutomationAccountName 'myAutomationAccount' -AzureVMName 'DscVm'
```

Specify configuration mode settings

Use the [Register-AzAutomationDscNode](#) cmdlet to register a VM as a managed node and specify configuration properties. For example, you can specify that the state of the machine is to be applied only once by specifying `ApplyOnly` as the value of the `ConfigurationMode` property. State Configuration doesn't try to apply the configuration after the initial check.

```
Register-AzAutomationDscNode -ResourceGroupName 'MyResourceGroup' -AutomationAccountName  
'myAutomationAccount' -AzureVMName 'DscVm' -ConfigurationMode 'ApplyOnly'
```

You can also specify how often DSC checks the configuration state by using the `ConfigurationModeFrequencyMins` property. For more information about DSC configuration settings, see [Configuring the Local Configuration Manager](#).

```
# Run a DSC check every 60 minutes  
Register-AzAutomationDscNode -ResourceGroupName 'MyResourceGroup' -AutomationAccountName  
'myAutomationAccount' -AzureVMName 'DscVm' -ConfigurationModeFrequencyMins 60
```

Assign a node configuration to a managed node

Now we can assign the compiled node configuration to the VM we want to configure.

```
# Get the ID of the DSC node  
$node = Get-AzAutomationDscNode -ResourceGroupName 'MyResourceGroup' -AutomationAccountName  
'myAutomationAccount' -Name 'DscVm'  
  
# Assign the node configuration to the DSC node  
Set-AzAutomationDscNode -ResourceGroupName 'MyResourceGroup' -AutomationAccountName 'myAutomationAccount' -  
NodeConfigurationName 'TestConfig.WebServer' -NodeId $node.Id
```

This assigns the node configuration named `TestConfig.WebServer` to the registered DSC node `DscVm`. By default, the DSC node is checked for compliance with the node configuration every 30 minutes. For information about how to change the compliance check interval, see [Configuring the Local Configuration Manager](#).

Check the compliance status of a managed node

You can get reports on the compliance status of a managed node using the `Get-AzAutomationDscNodeReport` cmdlet.

```
# Get the ID of the DSC node  
$node = Get-AzAutomationDscNode -ResourceGroupName 'MyResourceGroup' -AutomationAccountName  
'myAutomationAccount' -Name 'DscVm'  
  
# Get an array of status reports for the DSC node  
$reports = Get-AzAutomationDscNodeReport -ResourceGroupName 'MyResourceGroup' -AutomationAccountName  
'myAutomationAccount' -NodeId $node.Id  
  
# Display the most recent report  
$reports[0]
```

Next steps

- To get started, see [Get started with Azure Automation State Configuration](#).
- To learn how to enable nodes, see [Enable Azure Automation State Configuration](#).
- To learn about compiling DSC configurations so that you can assign them to target nodes, see [Compile DSC configurations in Azure Automation State Configuration](#).
- To see an example of using Azure Automation State Configuration in a continuous deployment pipeline, see [Set up continuous deployment with Chocolatey](#).
- For pricing information, see [Azure Automation State Configuration pricing](#).
- For a PowerShell cmdlet reference, see [Az.Automation](#).

Compose DSC configurations

3/5/2021 • 2 minutes to read • [Edit Online](#)

When you need to manage resource with more than a single desired state configuration (DSC), the best path is to use [composite resources](#). A composite resource is a nested and parameterized configuration being used as a DSC resource within another configuration. Use of composite resources allows you to create complex configurations while allowing the underlying composite resources to be individually managed and built.

Azure Automation enables the [import and compilation of composite resources](#). Once you've imported composite resources into your Automation account, you can use Azure Automation State Configuration through the **State Configuration (DSC)** feature in the Azure portal.

Compose a configuration

Before you can assign a configuration made from composite resources in the Azure portal, you must compose the configuration. Composition uses **Compose configuration** on the State Configuration (DSC) page while on either the **Configurations** or the **Compiled configurations** tab.

1. Sign in to the [Azure portal](#).
2. On the left, click **All resources** and then the name of your Automation account.
3. On the Automation account page, select **State configuration (DSC)** under **Configuration Management**.
4. On the State configuration (DSC) page, click either the **Configurations** or the **Compiled configurations** tab, then click **Compose configuration** in the menu at the top of the page.
5. On the **Basics** step, provide the new configuration name (required) and click anywhere on the row of each composite resource that you want to include in your new configuration, then click **Next** or click the **Source code** step. For the following steps, we selected `PSExecutionPolicy` and `RenameAndDomainJoin` composite resources.

Home > automation - State configuration (DSC) > Compose configuration

Compose configuration

/subscriptions/mySubscriptionNum/resourceGroups/ContosoVMs/providers/Microsoft.Automation/automationAccounts/automation

[Basics](#) [Source code](#) [Parameters](#)

Create a new configuration by using pre-defined configuration blocks called composites.

Give a name for the new configuration and select the desired composites.

* Configuration Name 

Composite resources		Module name	
<input type="text" value="Search composite resources"/>		2 selected	
COMPOSITE RESOURCES	PARAMETERS	DESCRIPTION	MODULE
<input type="checkbox"/> GroupSet	2 required, 5 optional	Provides a mechanism to manage local groups on the target node. Use this resource when you want to add and/or remove the same list of members to more than one group.	PSDscResources 2.9.0.0
<input type="checkbox"/> ProcessSet	2 required, 7 optional	A composite DSC resource to configure a set of similar WindowsProcess resources.	PSDscResources 2.9.0.0
<input checked="" type="checkbox"/> PSExecutionPolicy	1 required, 2 optional	[Community Supported] Allows to set the desired PowerShell execution policy.	StateConfigCompositeResource
<input type="checkbox"/> RemoteDesktopSettings	2 required, 3 optional	[Community Supported] Configure the remote desktop setting for the machine.	StateConfigCompositeResource
<input checked="" type="checkbox"/> RenameAndDomainJoin	1 required, 5 optional	[Community Supported] Allows you to configure a computer modifying its name or Active Directory domain or workgroup membership.	StateConfigCompositeResource
<input type="checkbox"/> ServiceSet	5 required, 3 optional	A composite DSC resource to configure a set of similar Service resources.	PSDscResources 2.9.0.0
<input type="checkbox"/> WindowsFeatureSet	2 required, 6 optional	A composite DSC resource to configure a set of similar WindowsFeature resources.	PSDscResources 2.9.0.0
<input type="checkbox"/> WindowsOptionalFeatureSet	3 required, 5 optional	A composite DSC resource to configure a set of similar WindowsOptionalFeature resources.	PSDscResources 2.9.0.0

- The **Source code** step shows what the composed configuration of the selected composite resources looks like. You can see the merging of all parameters and how they are passed to the composite resource. When you are done reviewing the new source code, click **Next** or click the **Parameters** step.

Home > automation - State configuration (DSC) > Compose configuration

Compose configuration

/subscriptions/mySubscriptionNum/resourceGroups/ContosoVMs/providers/Microsoft.Automation/automationAccounts/automation

[Basics](#) [Source code](#) [Parameters](#)

The following PowerShell script will define the state of your VM's configuration as code:

```
Configuration MyCompositeConfig {
    param(
        [Parameter(Mandatory=$true)]
        [ValidateNotNullOrEmpty()]
        [String] $PSExecutionPolicy_PSExecutionPolicy,
```

[Next](#)

- On the **Parameters** step, the parameter for each composite resource is exposed so that values can be provided. If a parameter has a description, it is displayed next to the parameter field. If a parameter is of **PSCredential** type, the dropdown provides a list of **Credential** objects in the current Automation account. A **+ Add a credential** option is also available. Once all required parameters have been provided, click **Save and compile**.

The screenshot shows the 'Compose configuration' blade in the Azure portal. At the top, the path is 'Home > automation - State configuration (DSC) > Compose configuration'. The main area contains several configuration parameters:

- PSExecutionPolicy_PsDscRunAsCredential: Set to [PSCredential].
- PSExecutionPolicy_DependsOn: An empty list.
- RENAMEANDDOMAINJOIN: An empty list.
- RenameAndDomainJoin_Credential: A dropdown menu with the option '+ Add a credential'. To its right, a tooltip explains: 'Specify the name of the Automation Credential to use when joining the domain.'

At the bottom is a blue 'Save and compile' button.

Submit the configuration for compilation

Once the new configuration is saved, it is submitted for compilation. You can view the status of the compilation job like you do with any imported configuration. For more information, see [View a compilation job](#).

When compilation has completed successfully, the new configuration appears in the **Compiled configurations** tab. Then you can assign the configuration to a managed node, using the steps in [Reassigning a node to a different node configuration](#).

Next steps

- To learn how to enable nodes, see [Enable Azure Automation State Configuration](#).
- To learn about compiling DSC configurations so that you can assign them to target nodes, see [Compile DSC configurations in Azure Automation State Configuration](#).
- To see an example of using Azure Automation State Configuration in a continuous deployment pipeline, see [Set up continuous deployment with Chocolatey](#).
- For pricing information, see [Azure Automation State Configuration pricing](#).
- For a PowerShell cmdlet reference, see [Az.Automation](#).

Compile DSC configurations in Azure Automation State Configuration

4/22/2021 • 7 minutes to read • [Edit Online](#)

You can compile Desired State Configuration (DSC) configurations in Azure Automation State Configuration in the following ways:

- Azure State Configuration compilation service
 - Beginner method with interactive user interface
 - Easily track job state
- Windows PowerShell
 - Call from Windows PowerShell on local workstation or build service
 - Integrate with development test pipeline
 - Provide complex parameter values
 - Work with node and non-node data at scale
 - Significant performance improvement

You can also use Azure Resource Manager templates with Azure Desired State Configuration (DSC) extension to push configurations to your Azure VMs. The Azure DSC extension uses the Azure VM Agent framework to deliver, enact, and report on DSC configurations running on Azure VMs. For compilation details using Azure Resource Manager templates, see [Desired State Configuration extension with Azure Resource Manager templates](#).

Compile a DSC configuration in Azure State Configuration

Portal

1. In your Automation account, click **State configuration (DSC)**.
2. Click on the **Configurations** tab, then click on the configuration name to compile.
3. Click **Compile**.
4. If the configuration has no parameters, you're prompted to confirm if you want to compile it. If the configuration has parameters, the **Compile Configuration** blade opens so that you can provide parameter values.
5. The Compilation Job page is opened so that you can track compilation job status. You can also use this page to track the node configurations (MOF configuration documents) placed on the Azure Automation State Configuration pull server.

Azure PowerShell

You can use [Start-AzAutomationDscCompilationJob](#) to start compiling with Windows PowerShell. The following sample code begins compilation of a DSC configuration called **SampleConfig**.

```
Start-AzAutomationDscCompilationJob -ResourceGroupName 'MyResourceGroup' -AutomationAccountName 'MyAutomationAccount' -ConfigurationName 'SampleConfig'
```

`Start-AzAutomationDscCompilationJob` returns a compilation job object that you can use to track job status. You can then use this compilation job object with [Get-AzAutomationDscCompilationJob](#) to determine the status of the compilation job, and [Get-AzAutomationDscCompilationJobOutput](#) to view its streams (output). The

following sample starts compilation of the SampleConfig configuration, waits until it has completed, and then displays its streams.

```
$CompilationJob = Start-AzAutomationDscCompilationJob -ResourceGroupName 'MyResourceGroup' -  
AutomationAccountName 'MyAutomationAccount' -ConfigurationName 'SampleConfig'  
  
while($null -eq $CompilationJob.EndTime -and $null -eq $CompilationJob.Exception)  
{  
    $CompilationJob = $CompilationJob | Get-AzAutomationDscCompilationJob  
    Start-Sleep -Seconds 3  
}  
  
$CompilationJob | Get-AzAutomationDscCompilationJobOutput -Stream Any
```

Declare basic parameters

Parameter declaration in DSC configurations, including parameter types and properties, works the same as in Azure Automation runbooks. See [Starting a runbook in Azure Automation](#) to learn more about runbook parameters.

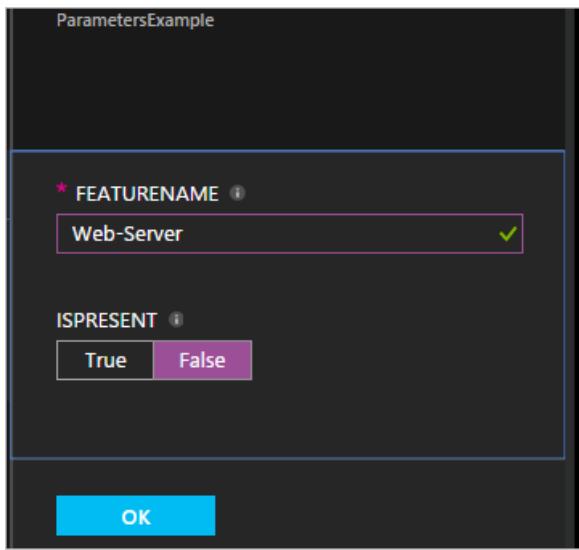
The following example uses `FeatureName` and `IsPresent` parameters to determine the values of properties in the `ParametersExample.sample` node configuration, generated during compilation.

```
Configuration ParametersExample  
{  
    param(  
        [Parameter(Mandatory=$true)]  
        [string] $FeatureName,  
  
        [Parameter(Mandatory=$true)]  
        [boolean] $IsPresent  
    )  
  
    $EnsureString = 'Present'  
    if($IsPresent -eq $false)  
    {  
        $EnsureString = 'Absent'  
    }  
  
    Node 'sample'  
    {  
        WindowsFeature ($FeatureName + 'Feature')  
        {  
            Ensure = $EnsureString  
            Name   = $FeatureName  
        }  
    }  
}
```

You can compile DSC configurations that use basic parameters in the Azure Automation State Configuration portal or with Azure PowerShell.

Portal

In the portal, you can enter parameter values after clicking **Compile**.



Azure PowerShell

PowerShell requires parameters in a `hashtable`, where the key matches the parameter name and the value equals the parameter value.

```
$Parameters = @{
    'FeatureName' = 'Web-Server'
    'IsPresent' = $False
}

Start-AzAutomationDscCompilationJob -ResourceGroupName 'MyResourceGroup' -AutomationAccountName
'MyAutomationAccount' -ConfigurationName 'ParametersExample' -Parameters $Parameters
```

For information about passing `PSCredential` objects as parameters, see [Credential assets](#).

Compile configurations containing composite resources in Azure Automation

The **Composite Resources** feature allows you to use DSC configurations as nested resources inside a configuration. This feature enables the application of multiple configurations to a single resource. See [Composite resources: Using a DSC configuration as a resource](#) to learn more about composite resources.

NOTE

So that configurations containing composite resources compile correctly, you must first import into Azure Automation any DSC resources that the composites rely upon. Adding a DSC composite resource is no different from adding any PowerShell module to Azure Automation. This process is documented in [Manage Modules in Azure Automation](#).

Manage ConfigurationData when compiling configurations in Azure Automation

`ConfigurationData` is a built-in DSC parameter that allows you to separate structural configuration from any environment-specific configuration while using PowerShell DSC. For more information, see [Separating "What" from "Where" in PowerShell DSC](#).

NOTE

When compiling in Azure Automation State Configuration, you can use `ConfigurationData` in Azure PowerShell but not in the Azure portal.

The following example DSC configuration uses `ConfigurationData` via the `$ConfigurationData` and `$AllNodes` keywords. You also need the [xWebAdministration module](#) for this example.

```

Configuration ConfigurationDataSample
{
    Import-DscResource -ModuleName xWebAdministration -Name MSFT_xWebsite

    Write-Verbose $ConfigurationData.NonNodeData.SomeMessage

    Node $AllNodes.Where{$_.Role -eq 'WebServer'}.NodeName
    {
        xWebsite Site
        {
            Name          = $Node.SiteName
            PhysicalPath = $Node.SiteContents
            Ensure        = 'Present'
        }
    }
}

```

You can compile the preceding DSC configuration with Windows PowerShell. The following script adds two node configurations to the Azure Automation State Configuration pull service: **ConfigurationDataSample.MyVM1** and **ConfigurationDataSample.MyVM3**.

```

$ConfigData = @{
    AllNodes = @(
        @{
            NodeName = 'MyVM1'
            Role = 'WebServer'
        },
        @{
            NodeName = 'MyVM2'
            Role = 'SQLServer'
        },
        @{
            NodeName = 'MyVM3'
            Role = 'WebServer'
        }
    )

    NonNodeData = @{
        SomeMessage = 'I love Azure Automation State Configuration and DSC!'
    }
}

Start-AzAutomationDscCompilationJob -ResourceGroupName 'MyResourceGroup' -AutomationAccountName
'MyAutomationAccount' -ConfigurationName 'ConfigurationDataSample' -ConfigurationData $ConfigData

```

Work with assets in Azure Automation during compilation

Asset references are the same in both Azure Automation State Configuration and runbooks. For more information, see the following:

- [Certificates](#)
- [Connections](#)
- [Credentials](#)
- [Variables](#)

Credential assets

DSC configurations in Azure Automation can reference Automation credential assets using the

`Get-AutomationPSCredential` cmdlet. If a configuration has a parameter that specifies a `PSCredential` object, use `Get-AutomationPSCredential` by passing the string name of an Azure Automation credential asset to the cmdlet to retrieve the credential. Then make use of that object for the parameter requiring the `PSCredential` object. Behind the scenes, the Azure Automation credential asset with that name is retrieved and passed to the

configuration. The example below shows this scenario in action.

Keeping credentials secure in node configurations (MOF configuration documents) requires encrypting the credentials in the node configuration MOF file. Currently you must give PowerShell DSC permission to output credentials in plain text during node configuration MOF generation. PowerShell DSC is not aware that Azure Automation encrypts the entire MOF file after its generation through a compilation job.

You can tell PowerShell DSC that it is okay for credentials to be outputted in plain text in the generated node configuration MOFs using configuration Data. You should pass `PSDscAllowPlainTextPassword = $true` via `ConfigurationData` for each node block name that appears in the DSC configuration and uses credentials.

The following example shows a DSC configuration that uses an Automation credential asset.

```
Configuration CredentialSample
{
    Import-DscResource -ModuleName PSDesiredStateConfiguration
    $Cred = Get-AutomationPSCredential 'SomeCredentialAsset'

    Node $AllNodes.NodeName
    {
        File ExampleFile
        {
            SourcePath      = '\\Server\share\path\file.ext'
            DestinationPath = 'C:\destinationPath'
            Credential     = $Cred
        }
    }
}
```

You can compile the preceding DSC configuration with PowerShell. The following PowerShell code adds two node configurations to the Azure Automation State Configuration pull server: `CredentialSample.MyVM1` and `CredentialSample.MyVM2`.

```
$ConfigData = @{
    AllNodes = @(
        @{
            NodeName = '*'
            PSDscAllowPlainTextPassword = $True
        },
        @{
            NodeName = 'MyVM1'
        },
        @{
            NodeName = 'MyVM2'
        }
    )
}

Start-AzAutomationDscCompilationJob -ResourceGroupName 'MyResourceGroup' -AutomationAccountName
'MyAutomationAccount' -ConfigurationName 'CredentialSample' -ConfigurationData $ConfigData
```

NOTE

When compilation is complete, you might receive the error message

```
The 'Microsoft.PowerShell.Management' module was not imported because the
'Microsoft.PowerShell.Management' snap-in was already imported.
```

You can safely ignore this message.

Compile your DSC configuration in Windows PowerShell

The process to compile DSC configurations in Windows PowerShell is included in the PowerShell DSC documentation [Write, Compile, and Apply a Configuration](#). You can execute this process from a developer workstation or within a build service, such as [Azure DevOps](#). You can then import the MOF files produced by compiling the configuration into the Azure State Configuration service.

Compiling in Windows PowerShell also provides the option to sign configuration content. The DSC agent verifies a signed node configuration locally on a managed node. Verification ensures that the configuration applied to the node comes from an authorized source.

You can also import node configurations (MOF files) that have been compiled outside of Azure. The import includes compilation from a developer workstation or in a service such as [Azure DevOps](#). This approach has multiple advantages, including performance and reliability.

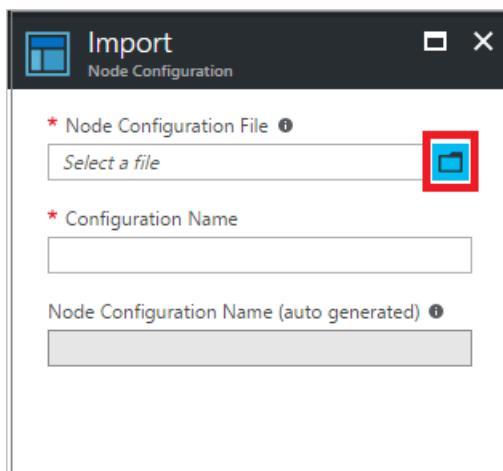
NOTE

A node configuration file must be no larger than 1 MB to allow Azure Automation to import it.

For more information about signing of node configurations, see [Improvements in WMF 5.1 - How to sign configuration and module](#).

Import a node configuration in the Azure portal

1. In your Automation account, click **State configuration (DSC)** under Configuration Management.
2. On the State configuration (DSC) page, click on the **Configurations** tab, then click **Add**.
3. On the Import page, click the folder icon next to the **Node Configuration File** field to browse for a node configuration MOF file on your local computer.



4. Enter a name in the **Configuration Name** field. This name must match the name of the configuration from which the node configuration was compiled.
5. Click **OK**.

Import a node configuration with Azure PowerShell

You can use the [Import-AzAutomationDscNodeConfiguration](#) cmdlet to import a node configuration into your Automation account.

```
Import-AzAutomationDscNodeConfiguration -AutomationAccountName 'MyAutomationAccount' -ResourceGroupName 'MyResourceGroup' -ConfigurationName 'MyNodeConfiguration' -Path 'C:\MyConfigurations\TestVM1.mof'
```

Next steps

- To get started, see [Get started with Azure Automation State Configuration](#).
- To learn about compiling DSC configurations so that you can assign them to target nodes, see [Compile DSC configurations in Azure Automation State Configuration](#).
- For a PowerShell cmdlet reference, see [Az.Automation](#).
- For pricing information, see [Azure Automation State Configuration pricing](#).
- For an example of using State Configuration in a continuous deployment pipeline, see [Set up continuous deployment with Chocolatey](#).

Remediate noncompliant Azure Automation State Configuration servers

9/1/2021 • 2 minutes to read • [Edit Online](#)

When servers are registered with Azure Automation State Configuration, the configuration mode is set to `ApplyOnly`, `ApplyAndMonitor`, or `ApplyAndAutoCorrect`. If the mode isn't set to `ApplyAndAutoCorrect`, servers that drift from a compliant state for any reason remain noncompliant until they're manually corrected.

Azure compute offers a feature named Run Command that allows customers to run scripts inside virtual machines. This document provides example scripts for this feature when manually correcting configuration drift.

Correct drift of Windows virtual machines using PowerShell

You can correct drift of Windows virtual machines using the `Run` command feature. See [Run PowerShell scripts in your Windows VM with Run command](#).

To force an Azure Automation State Configuration node to download the latest configuration and apply it, use the `Update-DscConfiguration` cmdlet.

```
Update-DscConfiguration -Wait -Verbose
```

Correct drift of Linux virtual machines

For Linux virtual machines, you don't have the option of using the `Run` command. You can only correct drift for these machines by repeating the registration process.

For Azure nodes, you can correct drift from the Azure portal or using Az module cmdlets. Details about this process are documented in [Enable a VM using Azure portal](#).

For hybrid nodes, you can correct drift using the Python scripts. See [Performing DSC operations from the Linux computer](#).

Next steps

- For a PowerShell cmdlet reference, see [Az.Automation](#).
- To see an example of using Azure Automation State Configuration in a continuous deployment pipeline, see [Set up continuous deployment with Chocolatey](#).

How to remove a configuration and node from Automation State Configuration

5/10/2021 • 2 minutes to read • [Edit Online](#)

This article covers how to unregister a node managed by Automation State Configuration, and safely remove a PowerShell Desired State Configuration (DSC) configuration from managed nodes. For both Windows and Linux nodes, you need to [unregister the node](#) and [delete the configuration](#). For Linux nodes only, you can optionally delete the DSC packages from the nodes as well. See [Remove the DSC package from a Linux node](#).

Unregister a node

If you no longer want a node to be managed by State Configuration (DSC), you can unregister it from the Azure portal or with Azure PowerShell using the following steps.

Unregistering a node from the service only sets the Local Configuration Manager settings so the node is no longer connecting to the service. This does not effect the configuration that's currently applied to the node, and leaves the related files in place on the node. You can optionally clean up those files. See [Delete a configuration](#).

Unregister in the Azure portal

1. Sign in to the [Azure portal](#).
2. Search for and select **Automation Accounts**.
3. On the **Automation Accounts** page, select your Automation account from the list.
4. From your Automation account, select **State configuration (DSC)** under **Configuration Management**.
5. On the **State configuration (DSC)** page, click the **Nodes** tab.
6. On the **Nodes** tab, select the name of the node you want to unregister.
7. On the pane for that node, click **Unregister**.

The screenshot shows the Azure portal interface for managing DSC nodes. At the top, there's a navigation bar with 'Microsoft Azure', a search bar, and user profile icons. Below that, a breadcrumb trail shows 'Home > Automation Accounts > MAIC-AA-Pri'. The main content area has a title 'SVR01' with a refresh icon. Underneath, there are two buttons: 'Assign node configuration' and 'Unregister', with 'Unregister' being the one highlighted with a red box. A collapsible section titled 'Essentials' lists various node details:

Resource group	: maic-rg	IP address	: 10.0.3.9
Id	: 12345678-1111-2222-3333-1234567891234	Account	: MAIC-AA-Pri
Last seen time	: 1/13/2021, 1:03 PM	Virtual machine	: SVR01
Configuration	: --	Node configurati...	: --
Registration ti...	: 2/6/2020, 10:54 AM	Status	: Unresponsive

Below this, there's a 'Reports' section which is currently empty, stating 'No reports found for this dsc node'.

Unregister using PowerShell

You can also unregister a node using the PowerShell cmdlet [Unregister-AzAutomationDscNode](#).

NOTE

If your organization still uses the deprecated AzureRM modules, you can use [Unregister-AzureRmAutomationDscNode](#).

Delete a configuration

When you're ready to remove an imported DSC configuration document (which is a Managed Object Format (MOF) or .mof file) that's assigned to one or more nodes, follow these steps.

Delete a configuration from the Azure portal

You can delete configurations for both Windows and Linux nodes from the Azure portal.

1. Sign in to the [Azure portal](#).
2. Search for and select **Automation Accounts**.
3. On the **Automation Accounts** page, select your Automation account from the list.
4. From your Automation account, select **State configuration (DSC)** under **Configuration Management**.
5. On the **State configuration (DSC)** page, click the **Configurations** tab, then select the name of the configuration you want to delete.

Configuration	Compiled Configuration Count	Last Modified	...
DomainControllerConfig	0	2/7/2020, 6:49 AM	...
HybridRunbookWorkerConfig	0	2/7/2020, 6:49 AM	...

6. On the configuration's detail page, click **Delete** to remove the configuration.

Resource group	(change) : InfraLab	Account	: AzureAutomation
Location	: East US 2	Subscription	(change) : Tailspin Toys
Subscription ID	: 12345678-1111-2222-3333-1234567891234	Status	: Published
Last published	: 2/22/2018, 1:22 PM	Configuration source	: View configuration source

Manually delete a configuration file from a node

If you don't want to use the Azure portal, you can manually delete the .mof configuration files as follows.

Delete a Windows configuration using PowerShell

To remove an imported DSC configuration document (.mof), use the [Remove-DscConfigurationDocument](#)

cmdlet.

Delete a Linux configuration

The configuration files are stored in `/etc/opt/omi/conf/dsc/configuration/`. Remove the .mof files in this directory to delete the node's configuration.

Remove the DSC package from a Linux node

This step is optional. Unregistering a Linux node from State Configuration (DSC) doesn't remove the DSC and OMI packages from the machine. Use the commands below to remove the packages as well as all logs and related data.

To find the package names and other relevant details, see the [PowerShell Desired State Configuration for Linux GitHub repository](#).

RPM-based systems

```
RPM -e <package name>
```

dpkg-based systems

```
dpkg -P <package name>
```

Next steps

- If you want to re-register the node, or register a new one, see [Register a VM to be managed by State Configuration](#).
- If you want to add the configuration back and recompile, see [Compile DSC configurations in Azure Automation State Configuration](#).

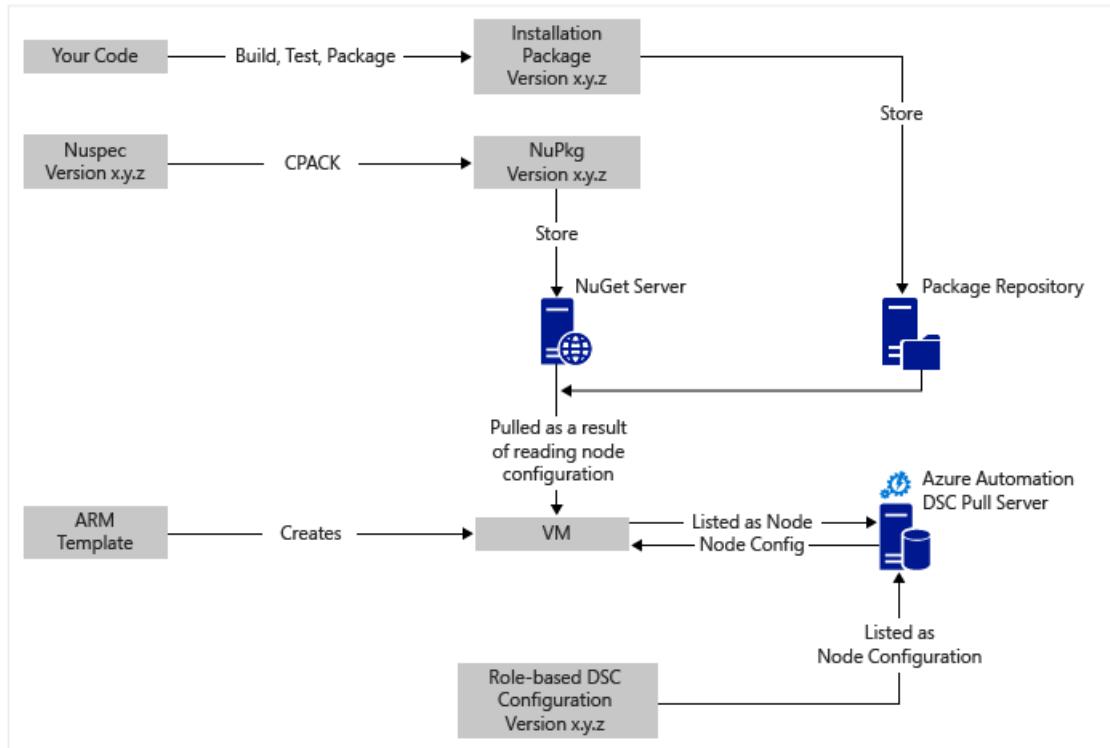
Set up continuous deployment with Chocolatey

6/14/2021 • 11 minutes to read • [Edit Online](#)

In a DevOps world, there are many tools to assist with various points in the continuous integration pipeline. Azure Automation [State Configuration](#) is a welcome new addition to the options that DevOps teams can employ.

Azure Automation is a managed service in Microsoft Azure that allows you to automate various tasks using runbooks, nodes, and shared resources, such as credentials, schedules, and global variables. Azure Automation State Configuration extends this automation capability to include PowerShell Desired State Configuration (DSC) tools. Here's a great [overview](#).

This article demonstrates how to set up Continuous Deployment (CD) for a Windows computer. You can easily extend the technique to include as many Windows computers as necessary in the role, for example, a website, and go from there to additional roles.



At a high level

There's quite a bit going on here, but fortunately it can be broken down into two main processes:

- Writing code and testing it, then creating and publishing installation packages for major and minor versions of the system.
- Creating and managing VMs that install and execute the code in the packages.

Once both of these core processes are in place, it's easy to automatically update the package on your VMs as new versions are created and deployed.

Component overview

Package managers such as [apt-get](#) are well known in the Linux world, but not so much in the Windows world. [Chocolatey](#) is such a thing, and Scott Hanselman's [blog](#) on the topic is a great introduction. In a nutshell, Chocolatey allows you to use the command line to install packages from a central repository onto a Windows

operating system. You can create and manage your own repository, and Chocolatey can install packages from any number of repositories that you designate.

[PowerShell DSC](#) is a PowerShell tool that allows you to declare the configuration that you want for a machine. For example, if you want Chocolatey installed, IIS installed, port 80 opened, and version 1.0.0 of your website installed, the DSC Local Configuration Manager (LCM) implements that configuration. A DSC pull server holds a repository of configurations for your machines. The LCM on each machine checks in periodically to see if its configuration matches the stored configuration. It can either report status or attempt to bring the machine back into alignment with the stored configuration. You can edit the stored configuration on the pull server to cause a machine or set of machines to come into alignment with the changed configuration.

A DSC resource is a module of code that has specific capabilities, such as managing networking, Active Directory, or SQL Server. The Chocolatey DSC Resource knows how to access a NuGet Server (among others), download packages, install packages, and so on. There are many other DSC Resources in the [PowerShell Gallery](#). You install these modules on your Azure Automation State Configuration pull server for use by your configurations.

Resource Manager templates provide a declarative way of generating your infrastructure, for example, networks, subnets, network security and routing, load balancers, NICs, VMs, and so on. Here's an [article](#) that compares the Resource Manager deployment model (declarative) with the Azure Service Management (ASM or classic) deployment model (imperative). This article includes a discussion of the core resource providers: compute, storage, and network.

One key feature of a Resource Manager template is its ability to install a VM extension into the VM as it's provisioned. A VM extension has specific capabilities, such as running a custom script, installing anti-virus software, and running a DSC configuration script. There are many other types of VM extensions.

Quick trip around the diagram

Starting at the top, you write your code, build it, test it, then create an installation package. Chocolatey can handle various types of installation packages, such as MSI, MSU, ZIP. And you have the full power of PowerShell to do the actual installation if Chocolatey's native capabilities aren't up to it. Put the package into some place reachable - a package repository. This usage example uses a public folder in an Azure blob storage account, but it can be anywhere. Chocolatey works natively with NuGet servers and a few others for management of package metadata. [This article](#) describes the options. The usage example uses NuGet. A Nuspec is metadata about your packages. The Nuspec information is compiled into a NuPkg and stored on a NuGet server. When your configuration requests a package by name and references a NuGet server, the Chocolatey DSC resource on the VM grabs the package and installs it. You can also request a specific version of a package.

In the bottom left of the picture, there's an Azure Resource Manager template. In this usage example, the VM extension registers the VM with the Azure Automation State Configuration pull server as a node. The configuration is stored in the pull server twice: once as plain text and once compiled as a MOF file. In the Azure portal, the MOF represents a node configuration, as opposed to a simple configuration. It's the artifact that's associated with a node so the node will know its configuration. Details below show how to assign the node configuration to the node.

Creating the Nuspec, compiling it, and storing it in a NuGet server is a small thing. And you're already managing VMs.

Taking the next step to continuous deployment requires setting up the pull server one time, registering your nodes with it one time, and creating and storing the initial configuration on the server. As packages are upgraded and deployed to the repository, you only have to refresh the configuration and node configuration on the pull server as needed.

If you're not starting with a Resource Manager template, that's fine. There are PowerShell commands to help you register your VMs with the pull server. For more information, see [Onboarding machines for management by](#)

About the usage example

The usage example in this article starts with a VM from a generic Windows Server 2012 R2 image from the Azure gallery. You can start from any stored image and then tweak from there with the DSC configuration. However, changing configuration that is baked into an image is much harder than dynamically updating the configuration using DSC.

You don't have to use a Resource Manager template and the VM extension to use this technique with your VMs. And your VMs don't have to be on Azure to be under CD management. All that's necessary is that Chocolatey be installed and the LCM configured on the VM so it knows where the pull server is.

When you update a package on a VM that's in production, you need to take that VM out of rotation while the update is installed. How you do this varies widely. For example, with a VM behind an Azure Load Balancer, you can add a Custom Probe. While updating the VM, have the probe endpoint return a 400. The tweak necessary to cause this change can be inside your configuration, as can the tweak to switch it back to returning a 200 once the update is complete.

Full source for this usage example is in [this Visual Studio project](#) on GitHub.

Step 1: Set up the pull server and Automation account

At an authenticated (`Connect-AzAccount`) PowerShell command line: (can take a few minutes while the pull server is set up)

```
New-AzResourceGroup -Name MY-AUTOMATION-RG -Location MY-RG-LOCATION-IN-QUOTES  
New-AzAutomationAccount -ResourceGroupName MY-AUTOMATION-RG -Location MY-RG-LOCATION-IN-QUOTES -Name MY-AUTOMATION-ACCOUNT
```

You can put your Automation account into any of the following regions (also known as locations): East US 2, South Central US, US Gov Virginia, West Europe, Southeast Asia, Japan East, Central India and Australia Southeast, Canada Central, North Europe.

Step 2: Make VM extension tweaks to the Resource Manager template

Details for VM registration (using the PowerShell DSC VM extension) provided in this [Azure Quickstart Template](#). This step registers your new VM with the pull server in the list of State Configuration Nodes. Part of this registration is specifying the node configuration to be applied to the node. This node configuration doesn't have to exist yet in the pull server, so it's fine that step 4 is where this is done for the first time. But here in Step 2 you do need to have decided the name of the node and the name of the configuration. In this usage example, the node is 'isvbox' and the configuration is 'ISVBoxConfig'. So the node configuration name (to be specified in DeploymentTemplate.json) is 'ISVBoxConfig.isvbox'.

Step 3: Add required DSC resources to the pull server

The PowerShell Gallery is instrumented to install DSC resources into your Azure Automation account. Navigate to the resource you want and click the "Deploy to Azure Automation" button.

The screenshot shows the PowerShell Gallery page for the 'xNetworking' module. At the top, there's a navigation bar with links for Home, Items, Publish, Statistics, Documentation, and Support, along with a search bar. Below the navigation is a large blue header with the PowerShell logo and the text 'PowerShell Gallery'. To the right of the header are 'Register' and 'Sign in' buttons. The main content area features a large blue icon of a greater-than sign (>). The module name 'xNetworking 5.4.0.0' is displayed in blue text. Below it, a description states 'Module with DSC Resources for Networking area'. There are four main action buttons: 'Inspect' (with a PS command: PS> Save-Module -Name xNetworking -Path <path>), 'Install' (with a PS command: PS> Install-Module -Name xNetworking), 'Deploy' (with a 'Deploy to Azure Automation' button), and 'Last published' (showing the date 2017-12-20). On the left side, there are statistics: 205,590 Downloads and 7,677 Downloads of 5.4.0.0.

Another technique recently added to the Azure portal allows you to pull in new modules or update existing modules. Click through the Automation account resource, the Assets tile, and finally the Modules tile. The Browse Gallery icon allows you to see the list of modules in the gallery, drill down into details and ultimately import into your Automation account. This is a great way to keep your modules up to date from time to time. And, the import feature checks dependencies with other modules to ensure nothing gets out of sync.

There's also a manual approach, used only once per resource, unless you want to upgrade it later. For more information on authoring PowerShell integration modules, see [Authoring Integration Modules for Azure Automation](#).

NOTE

The folder structure of a PowerShell integration module for a Windows computer is a little different from the folder structure expected by the Azure Automation.

1. Install [Windows Management Framework v5](#) (not needed for Windows 10).

2. Install the integration module.

```
Install-Module -Name MODULE-NAME`           <-grabs the module from the PowerShell Gallery
```

3. Copy the module folder from c:\Program Files\WindowsPowerShell\Modules\MODULE-NAME to a temporary folder.

4. Delete samples and documentation from the main folder.

5. Zip the main folder, naming the ZIP file with the name of the folder.

6. Put the ZIP file into a reachable HTTP location, such as blob storage in an Azure Storage account.

7. Run the following command.

```
New-AzAutomationModule `  
-ResourceGroupName MY-AUTOMATION-RG -AutomationAccountName MY-AUTOMATION-ACCOUNT `  
-Name MODULE-NAME -ContentLinkUri 'https://STORAGE-URI/CONTAINERNAME/MODULE-NAME.zip'
```

The included example implements these steps for cChoco and xNetworking.

Step 4: Add the node configuration to the pull server

There's nothing special about the first time you import your configuration into the pull server and compile. All later imports or compilations of the same configuration look exactly the same. Each time you update your package and need to push it out to production you do this step after ensuring the configuration file is correct - including the new version of your package. Here's the configuration file **ISVBoxConfig.ps1**:

```
Configuration ISVBoxConfig
{
    Import-DscResource -ModuleName cChoco
    Import-DscResource -ModuleName xNetworking

    Node 'isvbox' {

        cChocoInstaller installChoco
        {
            InstallDir = 'C:\choco'
        }

        WindowsFeature installIIS
        {
            Ensure = 'Present'
            Name   = 'Web-Server'
        }

        xFirewall WebFirewallRule
        {
            Direction      = 'Inbound'
            Name           = 'Web-Server-TCP-In'
            DisplayName    = 'Web Server (TCP-In)'
            Description    = 'IIS allow incoming web site traffic.'
            Enabled        = 'True'
            Action         = 'Allow'
            Protocol       = 'TCP'
            LocalPort      = '80'
            Ensure         = 'Present'
        }

        cChocoPackageInstaller trivialWeb
        {
            Name           = 'trivialweb'
            Version        = '1.0.0'
            Source         = 'MY-NUGET-V2-SERVER-ADDRESS'
            DependsOn     = '[cChocoInstaller]installChoco','[WindowsFeature]installIIS'
        }
    }
}
```

Here is the **New-ConfigurationScript.ps1** script (modified to use the Az module):

```

Import-AzAutomationDscConfiguration `

    -ResourceGroupName MY-AUTOMATION-RG -AutomationAccountName MY-AUTOMATION-ACCOUNT `

    -SourcePath C:\temp\AzureAutomationDsc\ISVBoxConfig.ps1 `

    -Published -Force

$jobData = Start-AzAutomationDscCompilationJob `

    -ResourceGroupName MY-AUTOMATION-RG -AutomationAccountName MY-AUTOMATION-ACCOUNT `

    -ConfigurationName ISVBoxConfig

$compilationJobId = $jobData.Id

Get-AzAutomationDscCompilationJob `

    -ResourceGroupName MY-AUTOMATION-RG -AutomationAccountName MY-AUTOMATION-ACCOUNT `

    -Id $compilationJobId

```

These steps result in a new node configuration named **ISVBoxConfig.isvbox** being placed on the pull server. The node configuration name is built as `configurationName.nodeName`.

Step 5: Create and maintain package metadata

For each package that you put into the package repository, you need a Nuspec that describes it. It must be compiled and stored on your NuGet server. This process is described [here](#).

You can use [MyGet.org](#) as a NuGet server. You can buy this service, but there is a free starter SKU. At [NuGet](#), you'll find instructions on installing your own NuGet server for your private packages.

Step 6: Tie it all together

Each time a version passes QA and is approved for deployment, the package is created, and nuspec and nupkg are updated and deployed to the NuGet server. The configuration (step 4) must also be updated to agree with the new version number. It must then be sent to the pull server and compiled.

From that point on, it's up to the VMs that depend on that configuration to pull the update and install it. Each of these updates is simple - just a line or two of PowerShell. For Azure DevOps, some of them are encapsulated in build tasks that can be chained together in a build. This [article](#) provides more details. This [GitHub repo](#) details the available build tasks.

Related articles

- [Azure Automation DSC overview](#)
- [Onboarding machines for management by Azure Automation DSC](#)

Next steps

- For an overview, see [Azure Automation State Configuration overview](#).
- To get started using the feature, see [Get started with Azure Automation State Configuration](#).
- To learn about compiling DSC configurations so that you can assign them to target nodes, see [Compile DSC configurations in Azure Automation State Configuration](#).
- For a PowerShell cmdlet reference, see [Az.Automation](#).
- For pricing information, see [Azure Automation State Configuration pricing](#).

Integrate Azure Automation State Configuration with Azure Monitor Logs

8/17/2021 • 8 minutes to read • [Edit Online](#)

Azure Automation State Configuration retains node status data for 30 days. You can send node status data to [Azure Monitor Logs](#) if you prefer to retain this data for a longer period. Compliance status is visible in the Azure portal or with PowerShell, for nodes and for individual DSC resources in node configurations.

Azure Monitor Logs provides greater operational visibility to your Automation State Configuration data and can help address incidents more quickly. With Azure Monitor Logs you can:

- Get compliance information for managed nodes and individual resources.
- Trigger an email or alert based on compliance status.
- Write advanced queries across your managed nodes.
- Correlate compliance status across Automation accounts.
- Use custom views and search queries to visualize your runbook results, runbook job status, and other related key indicators or metrics.

NOTE

This article was recently updated to use the term Azure Monitor logs instead of Log Analytics. Log data is still stored in a Log Analytics workspace and is still collected and analyzed by the same Log Analytics service. We are updating the terminology to better reflect the role of [logs in Azure Monitor](#). See [Azure Monitor terminology changes](#) for details.

Prerequisites

To start sending your Automation State Configuration reports to Azure Monitor Logs, you need:

- The PowerShell [Az Module](#) installed. Ensure you have the latest version. If necessary, run
`Update-Module -Name Az`.
- An Azure Automation account. For more information, see [An introduction to Azure Automation](#).
- A Log Analytics workspace. For more information, see [Azure Monitor Logs overview](#).
- At least one Azure Automation State Configuration node. For more information, see [Onboarding machines for management by Azure Automation State Configuration](#).
- The [xDscDiagnostics](#) module, version 2.7.0.0 or greater. For installation steps, see [Troubleshoot Azure Automation Desired State Configuration](#).

Set up integration with Azure Monitor Logs

To begin importing data from Azure Automation State Configuration into Azure Monitor Logs, complete the following steps. For steps using the Portal, see [Forward Azure Automation job data to Azure Monitor Logs](#).

1. From your machine, sign in to your Azure subscription with the PowerShell [Connect-AzAccount](#) cmdlet and follow the on-screen directions.

```

# Sign in to your Azure subscription
$sub = Get-AzSubscription -ErrorAction SilentlyContinue
if(-not($sub))
{
    Connect-AzAccount
}

# If you have multiple subscriptions, set the one to use
# Select-AzSubscription -SubscriptionId "<SUBSCRIPTIONID>"
```

- Provide an appropriate value for the variables `automationAccount` with the actual name of your Automation account, and `workspaceName` with the actual name of your Log Analytics workspace. Then execute the script.

```

$automationAccount = "automationAccount"
$law = "workspaceName"
```

- Get the resource ID of your Automation account by running the following PowerShell commands.

```

# Find the ResourceId for the Automation account
$AutomationResourceId = (Get-AzResource `-
    -ResourceType 'Microsoft.Automation/automationAccounts' | 
    WHERE {$_.Name -eq $automationAccount}).ResourceId
```

- Get the resource ID of your Log Analytics workspace by running the following PowerShell commands.

```

# Find the ResourceId for the Log Analytics workspace
$WorkspaceResourceId = (Get-AzResource `-
    -ResourceType 'Microsoft.OperationalInsights/workspaces' | 
    WHERE {$_.Name -eq $law}).ResourceId
```

- To configure diagnostic settings on the Automation account to forward DSC node status log data to Azure Monitor Logs, the following PowerShell cmdlet creates a diagnostic setting using that destination.

```

Set-AzDiagnosticSetting `-
    -ResourceId $AutomationResourceId `-
    -WorkspaceId $WorkspaceResourceId `-
    -Enabled $true `-
    -Category 'DscNodeStatus'
```

When you want to stop forwarding log data from Automation State Configuration to Azure Monitor Logs, run the following PowerShell cmdlet.

```

Set-AzDiagnosticSetting `-
    -ResourceId $AutomationResourceId `-
    -WorkspaceId $WorkspaceResourceId `-
    -Enabled $false `-
    -Category 'DscNodeStatus'
```

View the State Configuration logs

You can search the State Configuration logs for DSC operations by searching in Azure Monitor Logs. After you set up integration with Azure Monitor Logs for your Automation State Configuration data, navigate to your Automation account in the [Azure portal](#). Then under **Monitoring**, select **Logs**.

The screenshot shows the 'State configuration (DSC)' blade in the Azure portal. On the left, there's a sidebar with various account settings like Variables, Related Resources (Linked workspace, Event grid, Start/Stop VM), Account Settings (Properties, Keys, Pricing, Source control, Run as accounts), and Settings (Locks, Export template). The main area has tabs for Nodes, Configurations, Compiled configurations, and Gallery. A large circular summary chart shows 1 node overall, with 0 Failed, 0 Pending, 0 Not compliant, 0 In progress, 0 Unresponsive, and 1 Compliant. Below this is a table with columns Node, Status, Node configuration, Last seen, and Version. One row is shown for 'SVR01' which is 'Compliant'. At the bottom right, there's a note about VM DSC extension version > 2.70.

Close the **Queries** dialog box. The Log Search pane opens with a query region scoped to your Automation account resource. The records for DSC operations are stored in the `AzureDiagnostics` table. To find nodes that aren't compliant, type the following query.

```
AzureDiagnostics
| where Category == "DscNodeStatus"
| where OperationName contains "DscNodeStatusData"
| where ResultType != "Compliant"
```

Filtering details:

- Filter on `DscNodeStatusData` to return operations for each State Configuration node.
- Filter on `DscResourceStatusData` to return operations for each DSC resource called in the node configuration applied to that resource.
- Filter on `DscResourceStatusData` to return error information for any DSC resources that fail.

To learn more about constructing log queries to find data, see [Overview of log queries in Azure Monitor](#).

Send an email when a State Configuration compliance check fails

1. Return to your query created earlier.
2. Press on '+ New Alert Rule' button to start the alert creation flow.
3. In the query below, replace `NODENAME` with the actual name of the managed node, and then paste the revised query in the **Search query** text box:

```
AzureDiagnostics
| where Category == "DscNodeStatus"
| where NodeName_s == "NODENAME"
| where OperationName == "DscNodeStatusData"
| where ResultType == "Failed"
```

If you have set up logs from more than one Automation account or subscription to your workspace, you can group your alerts by subscription and Automation account. Derive the Automation account name from the `Resource` property in the log search results of the `DscNodeStatusData`.

4. Review [Create, view, and manage metric alerts using Azure Monitor](#) to complete the remaining steps.

Find failed DSC resources across all nodes

One advantage of using Azure Monitor Logs is that you can search for failed checks across nodes. To find all instances of DSC resources that have failed, use the following query:

```
AzureDiagnostics  
| where Category == "DscNodeStatus"  
| where OperationName == "DscResourceStatusData"  
| where ResultType == "Failed"
```

View historical DSC node status

To visualize your DSC node status history over time, you can use this query:

```
AzureDiagnostics  
| where ResourceProvider == "MICROSOFT.AUTOMATION"  
| where Category == "DscNodeStatus"  
| where ResultType != "started"  
| summarize count() by ResultType
```

This query displays a chart of the node status over time.

Azure Monitor Logs records

Azure Automation diagnostics create two categories of records in Azure Monitor Logs:

- Node status data (`DscNodeStatusData`)
- Resource status data (`DscResourceStatusData`)

DscNodeStatusData

PROPERTY	DESCRIPTION
TimeGenerated	Date and time when the compliance check ran.
OperationName	<code>DscNodeStatusData</code> .
ResultType	Value that indicates if the node is compliant.
NodeName_s	The name of the managed node.
NodeComplianceStatus_s	Status value that specifies if the node is compliant.
DscReportStatus	Status value indicating if the compliance check ran successfully.

PROPERTY	DESCRIPTION
ConfigurationMode	<p>The mode used to apply the configuration to the node. Possible values are:</p> <ul style="list-style-type: none"> • <code>ApplyOnly</code> : DSC applies the configuration and does nothing further unless a new configuration is pushed to the target node or when a new configuration is pulled from a server. After initial application of a new configuration, DSC doesn't check for drift from a previously configured state. DSC attempts to apply the configuration until it's successful before the <code>ApplyOnly</code> value takes effect. • <code>ApplyAndMonitor</code> : This is the default value. The LCM applies any new configurations. After initial application of a new configuration, if the target node drifts from the desired state, DSC reports the discrepancy in logs. DSC attempts to apply the configuration until it's successful before the <code>ApplyAndMonitor</code> value takes effect. • <code>ApplyAndAutoCorrect</code> : DSC applies any new configurations. After initial application of a new configuration, if the target node drifts from the desired state, DSC reports the discrepancy in logs, and then reapplies the current configuration.
HostName_s	The name of the managed node.
IPAddress	The IPv4 address of the managed node.
Category	<code>DscNodeStatus</code> .
Resource	The name of the Azure Automation account.
Tenant_g	GUID that identifies the tenant for the caller.
NodeId_g	GUID that identifies the managed node.
DscReportId_g	GUID that identifies the report.
LastSeenTime_t	Date and time when the report was last viewed.
ReportStartTime_t	Date and time when the report was started.
ReportEndTime_t	Date and time when the report completed.
NumberOfResources_d	The number of DSC resources called in the configuration applied to the node.
SourceSystem	The source system identifying how Azure Monitor Logs has collected the data. Always <code>Azure</code> for Azure diagnostics.
ResourceId	The resource identifier of the Azure Automation account.
ResultDescription	The resource description for this operation.

PROPERTY	DESCRIPTION
SubscriptionId	The Azure subscription ID (GUID) for the Automation account.
ResourceGroup	The name of the resource group for the Automation account.
ResourceProvider	MICROSOFT.AUTOMATION.
ResourceType	AUTOMATIONACCOUNTS.
CorrelationId	A GUID that is the correlation identifier of the compliance report.

DscResourceStatusData

PROPERTY	DESCRIPTION
TimeGenerated	Date and time when the compliance check ran.
OperationName	<code>DscResourceStatusData</code> .
ResultType	Whether the resource is compliant.
NodeName_s	The name of the managed node.
Category	<code>DscNodeStatus</code> .
Resource	The name of the Azure Automation account.
Tenant_g	GUID that identifies the tenant for the caller.
NodeId_g	GUID that identifies the managed node.
DscReportId_g	GUID that identifies the report.
DscResourceId_s	The name of the DSC resource instance.
Dsc resourceName_s	The name of the DSC resource.
DscResourceStatus_s	Whether the DSC resource is in compliance.
DscModuleName_s	The name of the PowerShell module that contains the DSC resource.
DscModuleVersion_s	The version of the PowerShell module that contains the DSC resource.
DscConfigurationName_s	The name of the configuration applied to the node.
ErrorCode_s	The error code if the resource failed.

PROPERTY	DESCRIPTION
ErrorMessage_s	The error message if the resource failed.
DscResourceDuration_d	The time, in seconds, that the DSC resource ran.
SourceSystem	How Azure Monitor Logs collected the data. Always Azure for Azure diagnostics.
ResourceId	The identifier of the Azure Automation account.
ResultDescription	The description for this operation.
SubscriptionId	The Azure subscription ID (GUID) for the Automation account.
ResourceGroup	The name of the resource group for the Automation account.
ResourceProvider	MICROSOFT.AUTOMATION.
ResourceType	AUTOMATIONACCOUNTS.
CorrelationId	GUID that is the correlation ID of the compliance report.

Next steps

- For an overview, see [Azure Automation State Configuration overview](#).
- To get started, see [Get started with Azure Automation State Configuration](#).
- To learn about compiling DSC configurations so that you can assign them to target nodes, see [Compile DSC configurations in Azure Automation State Configuration](#).
- For a PowerShell cmdlet reference, see [Az.Automation](#).
- For pricing information, see [Azure Automation State Configuration pricing](#).
- To see an example of using Azure Automation State Configuration in a continuous deployment pipeline, see [Set up continuous deployment with Chocolatey](#).
- To learn more about how to construct different search queries and review the Automation State Configuration logs with Azure Monitor Logs, see [Log searches in Azure Monitor Logs](#).
- To learn more about Azure Monitor Logs and data collection sources, see [Collecting Azure storage data in Azure Monitor Logs overview](#).

Work with Azure Desired State Configuration extension version history

4/2/2021 • 11 minutes to read • [Edit Online](#)

The Azure Desired State Configuration (DSC) VM [extension](#) is updated as-needed to support enhancements and new capabilities delivered by Azure, Windows Server, and the Windows Management Framework (WMF) that includes Windows PowerShell.

This article provides information about each version of the Azure DSC VM extension, what environments it supports, and comments and remarks on new features or changes.

Latest version

Version 2.83

- **Release date:**
 - February 2021
- **OS support:**
 - Windows Server 2019
 - Windows Server 2016
 - Windows Server 2012 R2
 - Windows Server 2012
 - Windows Server 2008 R2 SP1
 - Windows Client 7/8.1/10
 - Nano Server
- **WMF support:**
 - WMF 5.1
 - WMF 5.0 RTM
 - WMF 4.0 Update
 - WMF 4.0
- **Environment:**
 - Azure
 - Azure China Vianet 21
 - Azure Government
- **Remarks:** This release includes a fix for unsigned binaries with the Windows VM extension.

Version 2.80

- **Release date:**
 - September 26, Sep-2019 (Azure) | July 6, 2020 (Azure China Vianet 21) | July 20, 2020 (Azure Government)
- **OS support:**
 - Windows Server 2019
 - Windows Server 2016
 - Windows Server 2012 R2
 - Windows Server 2012
 - Windows Server 2008 R2 SP1

- Windows Client 7/8.1/10
- Nano Server
- **WMF support:**
 - WMF 5.1
 - WMF 5.0 RTM
 - WMF 4.0 Update
 - WMF 4.0
- **Environment:**
 - Azure
 - Azure China Vianet 21
 - Azure Government
- **Remarks:** No new features are included in this release.

Version 2.76

- **Release date:**
 - May 9, 2018 (Azure) | June 21, 2018 (Azure China Vianet 21, Azure Government)
- **OS support:**
 - Windows Server 2016
 - Windows Server 2012 R2
 - Windows Server 2012
 - Windows Server 2008 R2 SP1
 - Windows Client 7/8.1/10
 - Nano Server
- **WMF support:**
 - WMF 5.1
 - WMF 5.0 RTM
 - WMF 4.0 Update
 - WMF 4.0
- **Environment:**
 - Azure
 - Azure China Vianet 21
 - Azure Government
- **Remarks:** This version uses DSC as included in Windows Server 2016; for other Windows OSs, it installs the [Windows Management Framework 5.1](#) (installing WMF requires a reboot). For Nano Server, DSC role is installed on the VM.
- **New features:**
 - Improvement in extension metadata for substatus and other minor bug fixes.

Supported versions

WARNING

Versions 2.4 through 2.13 use WMF 5.0 Public Preview, whose signing certificates expired in August 2016. For more information about this issue, see the following [blog article](#).

Version 2.75

- **Release date:** March 5, 2018
- **OS support:** Windows Server 2016, Windows Server 2012 R2, Windows Server 2012, Windows Server

2008 R2 SP1, Windows Client 7/8.1/10, Nano Server

- **WMF support:** WMF 5.1, WMF 5.0 RTM, WMF 4.0 Update, WMF 4.0
- **Environment:** Azure
- **Remarks:** This version uses DSC as included in Windows Server 2016; for other Windows OSs, it installs the [Windows Management Framework 5.1](#) (installing WMF requires a reboot). For Nano Server, DSC role is installed on the VM.
- **New features:**
 - After GitHub's enforcement of TLS 1.2, you can't onboard a VM to Azure Automation State Configuration using DIY Resource Manager templates available on Azure Marketplace, or use DSC extension to retrieve any configurations hosted on GitHub. An error similar to the following while deploying the extension is returned:

```
{  
    "code": "DeploymentFailed",  
    "message": "At least one resource deployment operation failed. Please list deployment  
operations for details. Please see https://aka.ms/arm-debug for usage details.",  
    "details": [  
        {  
            "code": "Conflict",  
            "message": "{  
                \"status\": \"Failed\",  
                \"error\": {  
                    \"code\": \"ResourceDeploymentFailure\",  
                    \"message\": \"The resource operation completed with terminal provisioning  
state 'Failed'.\",  
                    \"details\": [ {  
                        \"code\": \"VMExtensionProvisioningError\",  
                        \"message\": \"VM has reported a failure when processing extension  
'Microsoft.PowerShell.DSC'.  
                            Error message: \\"\\\"The DSC Extension failed to execute: Error downloading  
                            https://github.com/Azure/azure-quickstart-templates/raw/master/dsc-extension-azure-automation-pullserver/UpdateLCMforAAPull.zip  
                            after 29 attempts: The request was aborted: Could not create SSL/TLS  
secure channel..\\nMore information about the failure can  
be found in the logs located under  
'C:\\Windows\\Azure\\\\Logs\\\\Plugins\\\\Microsoft.PowerShell.DSC\\\\2.74.0.0' on the  
VM.\\\".\""  
                    } ]  
                }  
            }  
        }]  
    }  
}
```

- In the new extension version, TLS 1.2 is now enforced. While deploying the extension, if you already specified `AutoUpgradeMinorVersion = true` in the Resource Manager template, the extension is autoupgraded to 2.75. For manual updates, specify `TypeHandlerVersion = 2.75` in your Resource Manager template.

Version 2.70 - 2.72

- **Release date:** November 13, 2017
- **OS support:** Windows Server 2016, Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2 SP1, Windows Client 7/8.1/10, Nano Server
- **WMF support:** WMF 5.1, WMF 5.0 RTM, WMF 4.0 Update, WMF 4.0
- **Environment:** Azure
- **Remarks:** This version uses DSC as included in Windows Server 2016; for other Windows OSs, it installs the [Windows Management Framework 5.1](#) (installing WMF requires a reboot). For Nano Server, DSC role is installed on the VM.
- **New features:**

- Bug fixes & improvements that simplify using Azure Automation State Configuration in the portal and with a Resource Manager template. For more information, see [Default Configuration Script](#) in the DSC extension documentation.

Version 2.26

- **Release date:** June 9, 2017
- **OS support:** Windows Server 2016, Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2 SP1, Windows Client 7/8.1/10, Nano Server
- **WMF support:** WMF 5.1, WMF 5.0 RTM, WMF 4.0 Update, WMF 4.0
- **Environment:** Azure
- **Remarks:** This version uses DSC as included in Windows Server 2016; for other Windows OSs, it installs the [Windows Management Framework 5.1](#) (installing WMF requires a reboot). For Nano Server, DSC role is installed on the VM.
- **New features:**
 - Telemetry improvements.

Version 2.25

- **Release date:** June 2, 2017
- **OS support:** Windows Server 2016, Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2 SP1, Windows Client 7/8.1/10, Nano Server
- **WMF support:** WMF 5.1, WMF 5.0 RTM, WMF 4.0 Update, WMF 4.0
- **Environment:** Azure
- **Remarks:** This version uses DSC as included in Windows Server 2016; for other Windows OSs, it installs the [Windows Management Framework 5.1](#) (installing WMF requires a reboot). For Nano Server, DSC role is installed on the VM.
- **New features:**
 - Several bug fixes and other minor improvements were added.

Version 2.24

- **Release date:** April 13, 2017
- **OS support:** Windows Server 2016, Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2 SP1, Nano Server
- **WMF support:** WMF 5.1, WMF 5.0 RTM, WMF 4.0 Update, WMF 4.0
- **Environment:** Azure
- **Remarks:** This version uses DSC as included in Windows Server 2016; for other Windows OSs, it installs the [Windows Management Framework 5.1](#) (installing WMF requires a reboot). For Nano Server, DSC role is installed on the VM.
- **New features:**
 - Exposes VM UUID & DSC Agent ID as extension metadata. Other minor improvements were added.

Version 2.23

- **Release date:** March 15, 2017
- **OS support:** Windows Server 2016, Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2 SP1, Nano Server
- **WMF support:** WMF 5.1, WMF 5.0 RTM, WMF 4.0 Update, WMF 4.0
- **Environment:** Azure
- **Remarks:** This version uses DSC as included in Windows Server 2016; for other Windows OSs, it installs the [Windows Management Framework 5.1](#) (installing WMF requires a reboot). For Nano Server, DSC role is installed on the VM.
- **New features:**

- Bug fixes and other improvements were added.

Version 2.22

- **Release date:** February 8, 2017
- **OS support:** Windows Server 2016, Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2 SP1, Nano Server
- **WMF support:** WMF 5.1, WMF 5.0 RTM, WMF 4.0 Update, WMF 4.0
- **Environment:** Azure
- **Remarks:** This version uses DSC as included in Windows Server 2016; for other Windows OSs, it installs the [Windows Management Framework 5.1](#) (installing WMF requires a reboot). For Nano Server, DSC role is installed on the VM.
- **New features:**
 - The DSC extension now supports WMF 5.1.
 - Minor other improvements were added.

Version 2.21

- **Release date:** December 2, 2016
- **OS support:** Windows Server 2016, Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2 SP1, Nano Server
- **WMF support:** WMF 5.1 Preview, WMF 5.0 RTM, WMF 4.0 Update, WMF 4.0
- **Environment:** Azure
- **Remarks:** This version uses DSC as included in Windows Server 2016; for other Windows OSs, it installs the [Windows Management Framework 5.0 RTM](#) (installing WMF requires a reboot). For Nano Server, DSC role is installed on the VM.
- **New features:**
 - The DSC extension is now available on Nano Server. This version primarily contains code changes for running the extension on Nano Server.
 - Minor other improvements were added.

Version 2.20

- **Release date:** August 2, 2016
- **OS support:** Windows Server 2016 Technical Preview, Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2 SP1
- **WMF support:** WMF 5.1 Preview, WMF 5.0 RTM, WMF 4.0 Update, WMF 4.0
- **Environment:** Azure
- **Remarks:** This version uses DSC as included in Windows Server 2016 Technical Preview; for other Windows OSs, it installs the [Windows Management Framework 5.0 RTM](#) (installing WMF requires a reboot).
- **New features:**
 - Support for WMF 5.1 Preview. When first published, this version was an optional upgrade and you had to specify Wmfversion = '5.1PP' in Resource Manager templates to install WMF 5.1 preview. Wmfversion = 'latest' still installs the [WMF 5.0 RTM](#). For more information on WMF 5.1 preview, see [this blog](#).
 - Minor other fixes and improvements were added.

Version 2.19

- **Release date:** June 3, 2016
- **OS support:** Windows Server 2016 Technical Preview, Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2 SP1
- **WMF support:** WMF 5.0 RTM, WMF 4.0 Update, WMF 4.0
- **Environment:** Azure, Azure China Vianet 21, Azure Government

- **Remarks:** This version uses DSC as included in Windows Server 2016 Technical Preview; for other Windows OSs, it installs the [Windows Management Framework 5.0 RTM](#) (installing WMF requires a reboot).
- **New features:**
 - The DSC extension is now available in Azure China Vianet 21. This version contains fixes for running the extension on Azure China Vianet 21.

Version 2.18

- **Release date:** June 3, 2016
- **OS support:** Windows Server 2016 Technical Preview, Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2 SP1
- **WMF support:** WMF 5.0 RTM, WMF 4.0 Update, WMF 4.0
- **Environment:** Azure
- **Remarks:** This version uses DSC as included in Windows Server 2016 Technical Preview; for other Windows OSs, it installs the [Windows Management Framework 5.0 RTM](#) (installing WMF requires a reboot).
- **New features:**
 - Make telemetry non-blocking when an error occurs during telemetry hotfix download (known Azure DNS issue) or during install.
 - Fix for the intermittent issue where extension stops processing configuration after a reboot. This was causing the DSC extension to remain in 'transitioning' state.
 - Minor other fixes and improvements were added.

Version 2.17

- **Release date:** April 26, 2016
- **OS support:** Windows Server 2016 Technical Preview, Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2 SP1
- **WMF support:** WMF 5.0 RTM, WMF 4.0 Update, WMF 4.0
- **Environment:** Azure
- **Remarks:** This version uses DSC as included in Windows Server 2016 Technical Preview; for other Windows OSs, it installs the [Windows Management Framework 5.0 RTM](#) (installing WMF requires a reboot).
- **New features:**
 - Support for WMF 4.0 Update. For more information on WMF 4.0 Update, see [this blog](#).
 - Retry logic on errors that occur during the DSC extension install or while applying a DSC configuration post extension install. As a part of this change, the extension will retry the installation if a previous install failed or re-enact a DSC configuration that had previously failed, for a maximum three times until it reaches the completion state (Success/Error) or if a new request comes. If the extension fails due to invalid user settings/user input, it does not retry. In this case, the extension needs to be invoked again with a new request and correct user settings.

NOTE

The DSC extension is dependent on the Azure VM agent for the retries. Azure VM agent invokes the extension with the last failed request until it reaches a success or error state.

Version 2.16

- **Release date:** April 21, 2016
- **OS support:** Windows Server 2016 Technical Preview, Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2 SP1

- **WMF support:** WMF 5.0 RTM, WMF 4.0
- **Environment:** Azure
- **Remarks:** This version uses DSC as included in Windows Server 2016 Technical Preview; for other Windows OSs, it installs the [Windows Management Framework 5.0 RTM](#) (installing WMF requires a reboot).
- **New features:**
 - Improvement in error handling and other minor bug fixes.
 - New property in DSC extension settings. `ForcePullAndApply` in AdvancedOptions is added to enable the DSC extension enact DSC configurations when the refresh mode is Pull (as opposed to the default Push mode). For more information about the DSC extension settings, refer to [this blog](#).

Version 2.15

- **Release date:** March 14, 2016
- **OS support:** Windows Server 2016 Technical Preview, Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2 SP1
- **WMF support:** WMF 5.0 RTM, WMF 4.0
- **Environment:** Azure
- **Remarks:** This version uses DSC as included in Windows Server 2016 Technical Preview; for other Windows OSs, it installs the [Windows Management Framework 5.0 RTM](#) (installing WMF requires a reboot).
- **New features:**
 - In extension version 2.14, changes to install WMF RTM were included. While upgrading from extension version 2.13.2.0 to 2.14.0.0, you may notice that some DSC cmdlets fail or your configuration fails with an error - 'No Instance found with given property values'. For more information, see the [DSC release notes](#). The workarounds for these issues have been added in 2.15 version.
 - If you already installed version 2.14 and are running into one of the above two issues, you need to perform these steps manually. In an elevated PowerShell session run the following commands:
 - `Remove-Item -Path $env:SystemRoot\system32\Configuration\DSCEngineCache.mof`
 - `mofcomp $env:windir\system32\wbem\DsccoreConfProv.mof`

Version 2.14

- **Release date:** February 25, 2016
- **OS support:** Windows Server 2016 Technical Preview, Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2 SP1
- **WMF support:** WMF 5.0 RTM, WMF 4.0
- **Environment:** Azure
- **Remarks:** This version uses DSC as included in Windows Server 2016 Technical Preview; for other Windows OSs, it installs the [Windows Management Framework 5.0 RTM](#) (installing WMF requires a reboot).
- **New features:**
 - Uses WMF RTM.
 - Enables data collection in order to improve the quality of the DSC extension. For more information, see this [blog article](#).
 - Provides an updated settings format for the extension in a Resource Manager template. For more information, see this [blog article](#).
 - Bug fixes and other enhancements.

Next steps

- For more information about PowerShell DSC, see [PowerShell documentation center](#).
- Examine the [Resource Manager template for the DSC extension](#).

- For other functionality and resources that you can manage with PowerShell DSC, browse the [PowerShell gallery](#).
- For details about passing sensitive parameters into configurations, see [Manage credentials securely with the DSC extension handler](#).

Troubleshoot Azure Automation State Configuration issues

5/25/2021 • 9 minutes to read • [Edit Online](#)

This article provides information on troubleshooting and resolving issues that arise while you compile or deploy configurations in Azure Automation State Configuration. For general information about the State Configuration feature, see [Azure Automation State Configuration overview](#).

Diagnose an issue

When you receive a compilation or deployment error for configuration, here are a few steps to help you diagnose the issue.

1. Ensure that your configuration compiles successfully on the local machine

Azure Automation State Configuration is built on PowerShell Desired State Configuration (DSC). You can find the documentation for the DSC language and syntax in the [PowerShell DSC Docs](#).

By compiling a DSC configuration on your local machine, you can discover and resolve common errors, such as:

- Missing modules.
- Syntax errors.
- Logic errors.

2. View DSC logs on your node

If your configuration compiles successfully, but fails when applied to a node, you can find detailed information in the DSC logs. For information about where to find these logs, see [Where are the DSC Event Logs](#).

The [xDscDiagnostics](#) module can assist you in parsing detailed information from the DSC logs. If you contact support, they require these logs to diagnose your issue.

You can install the `xDscDiagnostics` module on your local machine by following the instructions in [Install the stable version module](#).

To install the `xDscDiagnostics` module on your Azure machine, use [Invoke-AzVMRunCommand](#). You can also use the **Run command** option in the Azure portal by following the steps in [Run PowerShell scripts in your Windows VM with Run Command](#).

For information on using `xDscDiagnostics`, see [Using xDscDiagnostics to analyze DSC logs](#). See also [xDscDiagnostics Cmdlets](#).

3. Ensure that nodes and the Automation workspace have required modules

DSC depends on modules installed on the node. When you use Azure Automation State Configuration, import any required modules into your Automation account by following the steps in [Import Modules](#). Configurations can also have a dependency on specific versions of modules. For more information, see [Troubleshoot modules](#).

Scenario: A configuration with special characters can't be deleted from the portal

Issue

When you attempt to delete a DSC configuration from the portal, you see the following error:

An error occurred while deleting the DSC configuration '<name>'. Error-details: The argument configurationName with the value <name> is not valid. Valid configuration names can contain only letters, numbers, and underscores. The name must start with a letter. The length of the name must be between 1 and 64 characters.

Cause

This error is a temporary issue that's planned to be resolved.

Resolution

Use the [Remove-AzAutomationDscConfiguration](#) cmdlet to delete the configuration.

Scenario: Failed to register the DSC Agent

Issue

When [Set-DscLocalConfigurationManager](#) or another DSC cmdlet, you receive the error:

```
Registration of the Dsc Agent with the server
https://<location>-agentservice-prod-1.azure-automation.net/accounts/00000000-0000-0000-0000-000000000000
failed. The
underlying error is: Failed to register Dsc Agent with AgentId 00000000-0000-0000-0000-000000000000 with the
server htt
ps://<location>-agentservice-prod-1.azure-automation.net/accounts/00000000-0000-0000-0000-000000000000-
000000000000/Nodes(AgentId='00000000-0000-0000-0000-000000000000'). .
+ CategoryInfo          : InvalidResult: (root/Microsoft/...gurationManager:String) [], CimException
+ FullyQualifiedErrorId :
RegisterDscAgentCommandFailed,Microsoft.PowerShell.DesiredStateConfiguration.Commands.Re
gisterDscAgentCommand
+ PSComputerName         : <computerName>
```

Cause

This error is normally caused by a firewall, the machine being behind a proxy server, or other network errors.

Resolution

Verify that your machine has access to the proper endpoints for DSC and try again. For a list of ports and addresses needed, see [Network planning](#).

Scenario: Status reports return the response code Unauthorized

Issue

When you register a node with Azure Automation State Configuration, you receive one of the following error messages:

```
The attempt to send status report to the server https://<your Automation account
URL>/accounts/xxxxxxxxxxxxxxxxxxxxx/Nodes(AgentId='xxxxxxxxxxxxxxxxxxxxxx')/SendReport returned
unexpected response code Unauthorized.
```

```
VM has reported a failure when processing extension 'Microsoft.Powershell.DSC / Registration of the Dsc
Agent with the server failed.
```

Cause

This issue is caused by a bad or expired certificate. See [Re-register a node](#).

This issue might also be caused by a proxy configuration not allowing access to *.azure-automation.net. For

more information, see [Configuration of private networks](#).

Resolution

Use the following steps to reregister the failing DSC node.

Step 1: Unregister the node

1. In the Azure portal, go to **Home > Automation Accounts > (your Automation account) > State configuration (DSC)**.
2. Select **Nodes**, and select the node having trouble.
3. Select **Unregister** to unregister the node.

Step 2: Uninstall the DSC extension from the node

1. In the Azure portal, go to **Home > Virtual Machine > (failing node) > Extensions**.
2. Select **Microsoft.Powershell.DSC**, the PowerShell DSC extension.
3. Select **Uninstall** to uninstall the extension.

Step 3: Remove all bad or expired certificates from the node

On the failing node from an elevated PowerShell prompt, run these commands:

```
$certs = @()
$certs += dir cert:\localmachine\my | ?{$_._FriendlyName -like "DSC"}
$certs += dir cert:\localmachine\my | ?{$_._FriendlyName -like "DSC-OaaS Client Authentication"}
$certs += dir cert:\localmachine\CA | ?{$_._subject -like "CN=AzureDSCExtension*"}
""";"== DSC Certificates found: " + $certs.Count
$certs | FL ThumbPrint,FriendlyName,Subject
If (($certs.Count) -gt 0)
{
    ForEach ($Cert in $certs)
    {
        RD -LiteralPath ($Cert.Pspath)
    }
}
```

Step 4: Reregister the failing node

1. In the Azure portal, go to **Home > Automation Accounts > (your Automation account) > State configuration (DSC)**.
2. Select **Nodes**.
3. Select **Add**.
4. Select the failing node.
5. Select **Connect**, and select your desired options.

Scenario: Node is in failed status with a "Not found" error

Issue

The node has a report with Failed status and contains the error:

```
The attempt to get the action from server https://<url>//accounts/<account-id>/Nodes(AgentId=<agent-id>)/GetDscAction failed because a valid configuration <guid> cannot be found.
```

Cause

This error typically occurs when the node is assigned to a configuration name, for example, **ABC**, instead of a node configuration (MOF file) name, for example, **ABC.WebServer**.

Resolution

- Make sure that you're assigning the node with the node configuration name and not the configuration

name.

- You can assign a node configuration to a node by using the Azure portal or with a PowerShell cmdlet.
 - In the Azure portal, go to **Home > Automation Accounts > (your Automation account) > State configuration (DSC)**. Then select a node and select **Assign node configuration**.
 - Use the [Set-AzAutomationDscNode](#) cmdlet.

Scenario: No node configurations (MOF files) were produced when a configuration was compiled

Issue

Your DSC compilation job suspends with the error:

```
Compilation completed successfully, but no node configuration **.mof** files were generated.
```

Cause

When the expression following the `Node` keyword in the DSC configuration evaluates to `$null`, no node configurations are produced.

Resolution

Use one of the following solutions to fix the problem:

- Make sure that the expression next to the `Node` keyword in the configuration definition isn't evaluating to Null.
- If you're passing [ConfigurationData](#) when you compile the configuration, make sure that you're passing the values that the configuration expects from the configuration data.

Scenario: The DSC node report becomes stuck in the In Progress state

Issue

The DSC agent outputs:

```
No instance found with given property values
```

Cause

You've upgraded your Windows Management Framework (WMF) version and have corrupted Windows Management Instrumentation (WMI).

Resolution

Follow the instructions in [DSC known issues and limitations](#).

Scenario: Unable to use a credential in a DSC configuration

Issue

Your DSC compilation job suspended with the error:

```
System.InvalidOperationException error processing property 'Credential' of type <some resource name>:  
Converting and storing an encrypted password as plaintext is allowed only if PSDscAllowPlainTextPassword is  
set to true.
```

Cause

You've used a credential in a configuration but didn't provide proper `ConfigurationData` to set `PSDscAllowPlainTextPassword` to true for each node configuration.

Resolution

Make sure to pass in the proper `ConfigurationData` to set `PSDscAllowPlainTextPassword` to true for each node configuration that's mentioned in the configuration. See [Compiling DSC configurations in Azure Automation State Configuration](#).

Scenario: "Failure processing extension" error when enabling a machine from a DSC extension

Issue

When you enable a machine by using a DSC extension, a failure occurs that contains the error:

```
VM has reported a failure when processing extension 'Microsoft.PowerShell.DSC'. Error message: \"DSC Configuration 'RegistrationMetaConfigV2' completed with error(s). Following are the first few: Registration of the Dsc Agent with the server <url> failed. The underlying error is: The attempt to register Dsc Agent with Agent Id <ID> with the server <url> return unexpected response code BadRequest. .\".
```

Cause

This error typically occurs when the node is assigned a node configuration name that doesn't exist in the service.

Resolution

- Make sure that you're assigning the node with a name that exactly matches the name in the service.
- You can choose to not include the node configuration name, which results in enabling the node but not assigning a node configuration.

Scenario: "One or more errors occurred" error when registering a node by using PowerShell

Issue

When you register a node by using [Register-AzAutomationDSCNode](#) or [Register-AzureRMAutomationDSCNode](#), you receive the following error:

```
One or more errors occurred.
```

Cause

This error occurs when you try to register a node in a separate subscription from that used by the Automation account.

Resolution

Treat the cross-subscription node as though it's defined for a separate cloud, or on-premises. Register the node by using one of these options for enabling machines:

- Windows: [Physical/virtual Windows machines on-premises, or in a cloud other than Azure/AWS](#).
- Linux: [Physical/virtual Linux machines on-premises, or in a cloud other than Azure](#).

Scenario: "Provisioning has failed" error message

Issue

When you register a node, you see the error:

```
Provisioning has failed
```

Cause

This message occurs when there's an issue with connectivity between the node and Azure.

Resolution

Determine if your node is in a virtual private network (VPN) or has other issues connecting to Azure. See [Troubleshoot feature deployment issues](#).

Scenario: Failure with a general error when applying a configuration in Linux

Issue

When you apply a configuration in Linux, a failure occurs that contains the error:

```
This event indicates that failure happens when LCM is processing the configuration. ErrorCode is 1.  
ErrorDetail is The SendConfigurationApply function did not succeed.. ResourceId is [resource]name and  
SourceInfo is ::nnn::n::resource. ErrorMessage is A general error occurred, not covered by a more specific  
error code..
```

Cause

If the `/tmp` location is set to `noexec`, the current version of DSC fails to apply configurations.

Resolution

Remove the `noexec` option from the `/tmp` location.

Scenario: Node configuration names that overlap can result in a bad release

Issue

When you use a single configuration script to generate multiple node configurations and some node configuration names are subsets of other names, the compilation service can end up assigning the wrong configuration. This issue only occurs when you use a single script to generate configurations with configuration data per node, and only when the name overlap occurs at the beginning of the string. An example is a single configuration script used to generate configurations based on node data passed as a hashtable using cmdlets, and the node data includes servers named `server` and `1server`.

Cause

This is a known issue with the compilation service.

Resolution

The best workaround is to compile locally or in a CI/CD pipeline and upload the node configuration MOF files directly to the service. If compilation in the service is a requirement, the next best workaround is to split the compilation jobs so that there's no overlap in names.

Scenario: Gateway timeout error on DSC configuration upload

Issue

You receive a `GatewayTimeout` error when you upload a DSC configuration.

Cause

DSC configurations that take a long time to compile can cause this error.

Resolution

You can make your DSC configurations parse faster by explicitly including the `ModuleName` parameter for any [Import-DSCResource](#) calls.

Next steps

If you don't see your problem here or you can't resolve your issue, try one of the following channels for additional support:

- Get answers from Azure experts through [Azure Forums](#).
- Connect with [@AzureSupport](#), the official Microsoft Azure account for improving customer experience. Azure Support connects the Azure community to answers, support, and experts.
- File an Azure support incident. Go to the [Azure support site](#), and select **Get Support**.

Configure data based on Security Technical Information Guide (STIG)

11/2/2020 • 2 minutes to read • [Edit Online](#)

Applies To: Windows PowerShell 5.1

Creating configuration content for the first time can be challenging. In many cases, the goal is to automate configuration of servers following a "baseline" that hopefully aligns to an industry recommendation.

NOTE

This article refers to a solution that is maintained by the Open Source community. Support is only available in the form of GitHub collaboration, not from Microsoft.

Community project: PowerSTIG

A community project named [PowerSTIG](#) aims to resolve this issue by generating DSC content based on [public information](#) provided about STIG (Security Technical Implementation Guide),

Dealing with baselines is more complicated than it sounds. Many organizations need to [document exceptions](#) to rules and manage that data at scale. PowerSTIG addresses the problem by providing [Composite Resources](#) to address each area of the configuration rather than trying to address the entire range of settings in one large file.

Once the configurations have been generated, you can use the [DSC Configuration scripts](#) to generate MOF files and [upload the MOF files to Azure Automation](#). Then register your servers from either [on-premises](#) or [in Azure](#) to pull configurations.

To try out PowerSTIG, visit the [PowerShell Gallery](#) and download the solution or click "Project Site" to view the [documentation](#).

Next steps

- To understand PowerShell DSC, see [Windows PowerShell Desired State Configuration overview](#).
- Find out about PowerShell DSC resources in [DSC Resources](#).
- For details of Local Configuration Manager configuration, see [Configuring the Local Configuration Manager](#).

Configure data at scale for Azure Automation State Configuration

11/2/2020 • 2 minutes to read • [Edit Online](#)

Applies To: Windows PowerShell 5.1

Managing hundreds or thousands of servers can be a challenge. Customers have provided feedback that the most difficult aspect is actually managing [configuration data](#). Organizing information across logical constructs like location, type, and environment.

NOTE

This article refers to a solution that is maintained by the Open Source community. Support is only available in the form of GitHub collaboration, not from Microsoft.

Community project: Datum

A community maintained solution named [Datum](#) has been created to resolve this challenge. Datum builds on great ideas from other configuration management platforms and implements the same type of solution for PowerShell DSC. Information is [organized in to text files](#) based on logical ideas. Examples would be:

- Settings that should apply globally
- Settings that should apply to all servers in a location
- Settings that should apply to all database servers
- Individual server settings

This information is organized in the file format you prefer (JSON, Yaml, or PSD1). Then cmdlets are provided to generate configuration data files by [consolidating the information](#) from each file in to single view of a server or server role.

Once the data files have been generated, you can use them with [DSC Configuration scripts](#) to generate MOF files and [upload the MOF files to Azure Automation](#). Then register your servers from either [on-premises](#) or [in Azure](#) to pull configurations.

To try out Datum, visit the [PowerShell Gallery](#) and download the solution or click "Project Site" to view the [documentation](#).

Next steps

- To understand PowerShell DSC, see [Windows PowerShell Desired State Configuration overview](#).
- Find out about PowerShell DSC resources in [DSC Resources](#).
- For details of Local Configuration Manager configuration, see [Configuring the Local Configuration Manager](#).

Create configurations from existing servers

5/3/2021 • 2 minutes to read • [Edit Online](#)

Applies To: Windows PowerShell 5.1

Creating configurations from existing servers can be a challenging task. You might not want *all* settings, just those that you care about. Even then you need to know in what order the settings must be applied in order for the configuration to apply successfully.

NOTE

This article refers to a solution that is maintained by the Open Source community. Support is only available in the form of GitHub collaboration, not from Microsoft.

Community project: ReverseDSC

A community maintained solution named [ReverseDSC](#) has been created to work in this area starting SharePoint.

The solution builds on the [SharePointDSC resource](#) and extends it to orchestrate [gathering information](#) from existing servers running SharePoint. The latest version has multiple [extraction modes](#) to determine what level of information to include.

The result of using the solution is generating [Configuration Data](#) to be used with SharePointDSC configuration scripts.

Once the data files have been generated, you can use them with [DSC Configuration scripts](#) to generate MOF files and [upload the MOF files to Azure Automation](#). Then register your servers from either [on-premises](#) or [in Azure](#) to pull configurations.

To try out ReverseDSC, visit the [PowerShell Gallery](#) and download the solution or click "Project Site" to view the [documentation](#).

Next steps

- To understand PowerShell DSC, see [Windows PowerShell Desired State Configuration overview](#).
- Find out about PowerShell DSC resources in [DSC Resources](#).
- For details of Local Configuration Manager configuration, see [Configuring the Local Configuration Manager](#).

Convert configurations to composite resources

5/3/2021 • 2 minutes to read • [Edit Online](#)

Applies To: Windows PowerShell 5.1

Once you get started authoring configurations, you can quickly create "scenarios" that manage groups of settings. Examples would be:

- create a web server
- create a DNS server
- create a server that runs SharePoint
- configure a SQL cluster
- manage firewall settings
- manage password settings

If you are interested in sharing this work with others, the best option is to package the configuration as a [Composite Resource](#). Creating composite resources for the first time can be overwhelming.

NOTE

This article refers to a solution that is maintained by the Open Source community. Support is only available in the form of GitHub collaboration, not from Microsoft.

Community project: CompositeResource

A community maintained solution named [CompositeResource](#) has been created to resolve this challenge.

CompositeResource automates the process of creating a new module from your configuration. You start by [dot sourcing](#) the configuration script on your workstation (or build server) so it is loaded in memory. Next, rather than running the configuration to generate a MOF file, use the function provided by the CompositeResource module to automate a conversion. The cmdlet will load the contents of your configuration, get the list of parameters, and generate a new module with everything you need.

Once you have generated a module, you can increment the version and add release notes each time you make changes and publish it to your own [PowerShellGet repository](#).

Once you have created a composite resource module containing your configuration (or multiple configurations), you can use them in the [Composable Authoring Experience](#) in Azure, or add them to [DSC Configuration scripts](#) to generate MOF files and [upload the MOF files to Azure Automation](#). Then register your servers from either [on-premises](#) or [in Azure](#) to pull configurations. The latest update to the project has also published [runbooks](#) for Azure Automation to automate the process of importing configurations from the PowerShell Gallery.

To try out automating creation of composite resources for DSC, visit the [PowerShell Gallery](#) and download the solution or click "Project Site" to view the [documentation](#).

Next steps

- To understand PowerShell DSC, see [Windows PowerShell Desired State Configuration overview](#).
- Find out about PowerShell DSC resources in [DSC Resources](#).
- For details of Local Configuration Manager configuration, see [Configuring the Local Configuration Manager](#).

Change Tracking and Inventory overview

8/24/2021 • 10 minutes to read • [Edit Online](#)

This article introduces you to Change Tracking and Inventory in Azure Automation. This feature tracks changes in virtual machines hosted in Azure, on-premises, and other cloud environments to help you pinpoint operational and environmental issues with software managed by the Distribution Package Manager. Items that are tracked by Change Tracking and Inventory include:

- Windows software
- Linux software (packages)
- Windows and Linux files
- Windows registry keys
- Windows services
- Linux daemons

NOTE

To track Azure Resource Manager property changes, see the Azure Resource Graph [change history](#).

Change Tracking and Inventory makes use of [Azure Security Center File Integrity Monitoring \(FIM\)](#) to examines operating system and application files, and Windows Registry. While FIM monitors those entities, Change Tracking and Inventory natively tracks:

- Software changes
- Windows services
- Linux daemons

Enabling all features included in Change Tracking and Inventory might cause additional charges. Before proceeding, review [Automation Pricing](#) and [Azure Monitor Pricing](#).

Change Tracking and Inventory forwards data to Azure Monitor Logs, and this collected data is stored in a Log Analytics workspace. The File Integrity Monitoring (FIM) feature is available only when [Azure Defender for servers](#) is enabled. See [Azure Security Center Pricing](#) to learn more. FIM uploads data to the same Log Analytics workspace as the one created to store data from Change Tracking and Inventory. We recommend that you monitor your linked Log Analytics workspace to keep track of your exact usage. For more information about analyzing Azure Monitor Logs data usage, see [Manage usage and cost](#).

Machines connected to the Log Analytics workspace use the [Log Analytics agent](#) to collect data about changes to installed software, Windows services, Windows registry and files, and Linux daemons on monitored servers. When data is available, the agent sends it to Azure Monitor Logs for processing. Azure Monitor Logs applies logic to the received data, records it, and makes it available for analysis.

NOTE

Change Tracking and Inventory requires linking a Log Analytics workspace to your Automation account. For a definitive list of supported regions, see [Azure Workspace mappings](#). The region mappings don't affect the ability to manage VMs in a separate region from your Automation account.

As a service provider, you may have onboarded multiple customer tenants to [Azure Lighthouse](#). Azure

Lighthouse allows you to perform operations at scale across several Azure Active Directory (Azure AD) tenants at once, making management tasks like Change Tracking and Inventory more efficient across those tenants you're responsible for. Change Tracking and Inventory can manage machines in multiple subscriptions in the same tenant, or across tenants using [Azure delegated resource management](#).

Current limitations

Change Tracking and Inventory doesn't support or has the following limitations:

- Recursion for Windows registry tracking
- Network file systems
- Different installation methods
- *.exe files stored on Windows
- The **Max File Size** column and values are unused in the current implementation.
- If you are tracking file changes, it is limited to a file size of 5 MB or less.
- If you try to collect more than 2500 files in a 30-minute collection cycle, Change Tracking and Inventory performance might be degraded.
- If network traffic is high, change records can take up to six hours to display.
- If you modify a configuration while a machine or server is shut down, it might post changes belonging to the previous configuration.
- Collecting Hotfix updates on Windows Server 2016 Core RS3 machines.
- Linux daemons might show a changed state even though no change has occurred. This issue arises because of the way the `SvcRunLevels` data in the Azure Monitor [ConfigurationChange](#) table is written.

Limits

For limits that apply to Change Tracking and Inventory, see [Azure Automation service limits](#).

Supported operating systems

Change Tracking and Inventory is supported on all operating systems that meet Log Analytics agent requirements. See [supported operating systems](#) for a list of the Windows and Linux operating system versions that are currently supported by the Log Analytics agent.

To understand client requirements for TLS 1.2, see [TLS 1.2 for Azure Automation](#).

Python requirement

Change Tracking and Inventory only supports Python2. If your machine is using a distro that doesn't include Python 2 by default then you must install it. The following sample commands will install Python 2 on different distros.

- Red Hat, CentOS, Oracle: `yum install -y python2`
- Ubuntu, Debian: `apt-get install -y python2`
- SUSE: `zypper install -y python2`

The `python2` executable must be aliased to `python`.

Network requirements

Check [Azure Automation Network Configuration](#) for detailed information on the ports, URLs, and other networking details required for Change Tracking and Inventory.

Enable Change Tracking and Inventory

You can enable Change Tracking and Inventory in the following ways:

- From your [Automation account](#) for one or more Azure and non-Azure machines.
- Manually for non-Azure machines, including machines or servers registered with [Azure Arc-enabled servers](#). For hybrid machines, we recommend installing the Log Analytics agent for Windows by first connecting your machine to [Azure Arc-enabled servers](#), and then using Azure Policy to assign the [Deploy Log Analytics agent to Linux or Windows Azure Arc machines](#) built-in policy. If you plan to also monitor the machines with Azure Monitor for VMs, instead use the [Enable Azure Monitor for VMs](#) initiative.
- For a single Azure VM from the [Virtual machine page](#) in the Azure portal. This scenario is available for Linux and Windows VMs.
- For [multiple Azure VMs](#) by selecting them from the Virtual machines page in the Azure portal.

Tracking file changes

For tracking changes in files on both Windows and Linux, Change Tracking and Inventory uses MD5 hashes of the files. The feature uses the hashes to detect if changes have been made since the last inventory.

Tracking file content changes

Change Tracking and Inventory allows you to view the contents of a Windows or Linux file. For each change to a file, Change Tracking and Inventory stores the contents of the file in an [Azure Storage account](#). When you're tracking a file, you can view its contents before or after a change. The file content can be viewed either inline or side by side.

The screenshot shows the Azure portal interface for viewing file content changes. The URL in the address bar is [Home > Automation Accounts > TestAzureAuto - Change tracking > View File Content Changes](#). The page title is "View File Content Changes" for the file "/etc/hosts". At the top, there are two tabs: "Side by side" (which is selected) and "Inline". The content area shows two columns of code. The left column shows the original file content, and the right column shows the modified file content with changes highlighted in green. Line numbers are present on the left of both columns.

1 127.0.0.1 localhost	1 127.0.0.1 localhost
2	2
3	3 + 5.6.7.8 badhost
4	4 +
5	5
6	6
7 # The following lines are desirable for IPv6 capable hosts	7 # The following lines are desirable for IPv6 capable hosts
8 ::1 ip6-localhost ip6-loopback	8 ::1 ip6-localhost ip6-loopback
9 fe00::0 ip6-localnet	9 fe00::0 ip6-localnet
10 ff00::0 ip6-mcastprefix	10 ff00::0 ip6-mcastprefix
11 ff02::1 ip6-allnodes	11 ff02::1 ip6-allnodes
12	12
13 ff02::2 ip6-allrouters	13 ff02::2 ip6-allrouters
14	14
15 ff02::3 ip6-allhosts	15 ff02::3 ip6-allhosts
16	16
17	17
18	18
19	19
20	20
21	21

Tracking of registry keys

Change Tracking and Inventory allows monitoring of changes to Windows registry keys. Monitoring allows you to pinpoint extensibility points where third-party code and malware can activate. The following table lists preconfigured (but not enabled) registry keys. To track these keys, you must enable each one.

REGISTRY KEY	PURPOSE
HKEY\LOCAL\MACHINE\Software\Microsoft\Windows\CurrentVersion\Run\Scripts\Startup	Monitors scripts that run at startup.
HKEY\LOCAL\MACHINE\Software\Microsoft\Windows\CurrentVersion\Run\Scripts\Shutdown	Monitors scripts that run at shutdown.
HKEY\LOCAL\MACHINE\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Run\keys that are loaded before the user signs in to the Windows account. The key is used for 32-bit applications running on 64-bit computers.	Monitors keys that are loaded before the user signs in to the Windows account. The key is used for 32-bit applications running on 64-bit computers.
HKEY\LOCAL\MACHINE\SOFTWARE\Microsoft\Active Setup\Installed Components	Monitors changes to application settings.
HKEY\LOCAL\MACHINE\Software\Classes\Directory\ShellEx\ContextMenuHandlers	Monitors context menu handlers that hook directly into Windows Explorer and usually run in-process with explorer.exe.
HKEY\LOCAL\MACHINE\Software\Classes\Directory\Shellex\ContextMenuHandlers	Monitors context menu handlers that hook directly into Windows Explorer and usually run in-process with explorer.exe.
HKEY\LOCAL\MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce\shellkey registrations	Monitors for new overlay handler registrations.
HKEY\LOCAL\MACHINE\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\RunOnce\shellkey registrations for 32-bit	Monitors for new overlay handler registration for 32-bit applications running on 64-bit computers.
HKEY\LOCAL\MACHINE\Software\Microsoft\Windows\CurrentVersion\Internet Helper Objects	Monitors for new browser helper object plugins for Internet Explorer. Used to access the Document Object Model (DOM) of the current page and to control navigation.
HKEY\LOCAL\MACHINE\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Internet Helper Objects	Monitors for new browser helper object plugins for Internet Explorer. Used to access the Document Object Model (DOM) of the current page and to control navigation for 32-bit applications running on 64-bit computers.
HKEY\LOCAL\MACHINE\Software\Microsoft\Internet Explorer\Extensions	Monitors for new Internet Explorer extensions, such as custom tool menus and custom toolbar buttons.
HKEY\LOCAL\MACHINE\Software\Wow6432Node\Microsoft\Internet Explorer\Extensions	Monitors for new Internet Explorer extensions, such as custom tool menus and custom toolbar buttons for 32-bit applications running on 64-bit computers.
HKEY\LOCAL\MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Drivers32	Monitors 32-bit drivers associated with wavemapper, wave1 and wave2, msacm.imaadpcm, .msadpcm, .msgsm610, and vidc. Similar to the [drivers] section in the system.ini file.
HKEY\LOCAL\MACHINE\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Drivers32	Monitors 32-bit drivers associated with wavemapper, wave1 and wave2, msacm.imaadpcm, .msadpcm, .msgsm610, and vidc for 32-bit applications running on 64-bit computers. Similar to the [drivers] section in the system.ini file.
HKEY\LOCAL\MACHINE\System\CurrentControlSet\Control\Session Manager\KnownDlls	Monitors the list of known or commonly used system DLLs. Monitoring prevents people from exploiting weak application directory permissions by dropping in Trojan horse versions of system DLLs.

REGISTRY KEY	PURPOSE
HKEY\LOCAL\MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Notify	Monitors the list of packages that can receive event notifications from winlogon.exe , the interactive logon support model for Windows.

Recursion support

Change Tracking and Inventory supports recursion, which allows you to specify wildcards to simplify tracking across directories. Recursion also provides environment variables to allow you to track files across environments with multiple or dynamic drive names. The following list includes common information you should know when configuring recursion:

- Wildcards are required for tracking multiple files.
- You can use wildcards only in the last segment of a file path, for example, `c:\folder\file*` or `/etc/* .conf`.
- If an environment variable has an invalid path, validation succeeds but the path fails during execution.
- You should avoid general path names when setting the path, as this type of setting can cause too many folders to be traversed.

Change Tracking and Inventory data collection

The next table shows the data collection frequency for the types of changes supported by Change Tracking and Inventory. For every type, the data snapshot of the current state is also refreshed at least every 24 hours.

CHANGE TYPE	FREQUENCY
Windows registry	50 minutes
Windows file	30 minutes
Linux file	15 minutes
Windows services	10 seconds to 30 minutes Default: 30 minutes
Linux daemons	5 minutes
Windows software	30 minutes
Linux software	5 minutes

The following table shows the tracked item limits per machine for Change Tracking and Inventory.

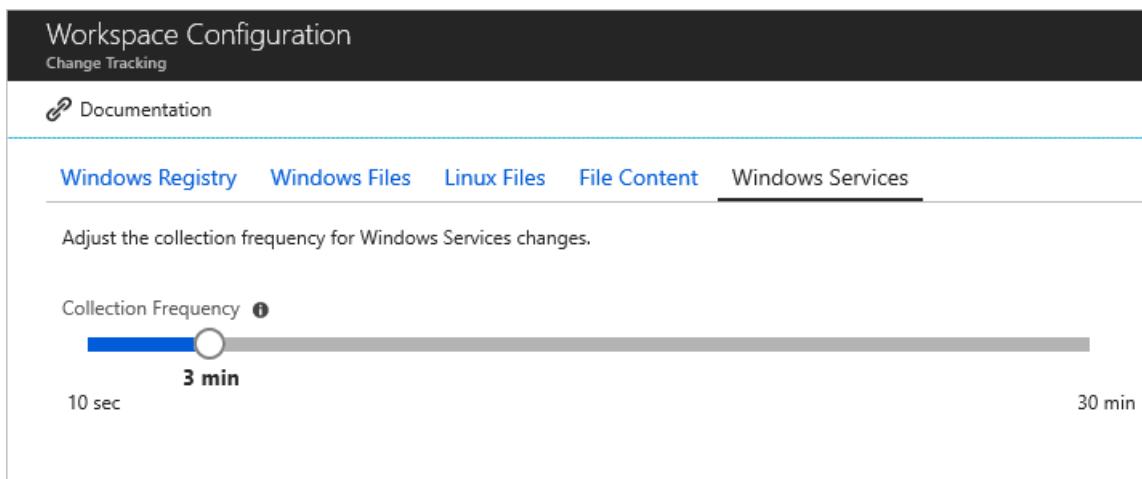
RESOURCE	LIMIT
File	500
Registry	250
Windows software (not including hotfixes)	250

RESOURCE	LIMIT
Linux packages	1250
Services	250
Daemons	250

The average Log Analytics data usage for a machine using Change Tracking and Inventory is approximately 40 MB per month, depending on your environment. With the Usage and Estimated Costs feature of the Log Analytics workspace, you can view the data ingested by Change Tracking and Inventory in a usage chart. Use this data view to evaluate your data usage and determine how it affects your bill. See [Understand your usage and estimate costs](#).

Windows services data

The default collection frequency for Windows services is 30 minutes. You can configure the frequency using a slider on the **Windows services** tab under **Edit Settings**.



To optimize performance, the Log Analytics agent only tracks changes. Setting a high threshold might miss changes if the service returns to its original state. Setting the frequency to a smaller value allows you to catch changes that might be missed otherwise.

NOTE

While the agent can track changes down to a 10-second interval, the data still takes a few minutes to display in the Azure portal. Changes that occur during the time to display in the portal are still tracked and logged.

Support for alerts on configuration state

A key capability of Change Tracking and Inventory is alerting on changes to the configuration state of your hybrid environment. Many useful actions are available to trigger in response to alerts. For example, actions on Azure functions, Automation runbooks, webhooks, and the like. Alerting on changes to the `c:\windows\system32\drivers\etc\hosts` file for a machine is one good application of alerts for Change Tracking and Inventory data. There are many more scenarios for alerting as well, including the query scenarios defined in the next table.

QUERY	DESCRIPTION
ConfigurationChange where ConfigChangeType == "Files" and FileSystemPath contains " c\windows\system32\drivers\"	Useful for tracking changes to system-critical files.
ConfigurationChange where FieldsChanged contains "FileContentChecksum" and FileSystemPath == "c\windows\system32\drivers\etc\hosts"	Useful for tracking modifications to key configuration files.
ConfigurationChange where ConfigChangeType == "WindowsServices" and SvcName contains "w3svc" and SvcState == "Stopped"	Useful for tracking changes to system-critical services.
ConfigurationChange where ConfigChangeType == "Daemons" and SvcName contains "ssh" and SvcState!= "Running"	Useful for tracking changes to system-critical services.
ConfigurationChange where ConfigChangeType == "Software" and ChangeCategory == "Added"	Useful for environments that need locked-down software configurations.
ConfigurationData where SoftwareName contains "Monitoring Agent" and CurrentVersion!= "8.0.11081.0"	Useful for seeing which machines have outdated or noncompliant software version installed. This query reports the last reported configuration state, but doesn't report changes.
ConfigurationChange where RegistryKey == @ "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\QualityCompat"	Useful for tracking changes to crucial antivirus keys.
ConfigurationChange where RegistryKey contains @ "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy"	Useful for tracking changes to firewall settings.

Next steps

- To enable from an Automation account, see [Enable Change Tracking and Inventory from an Automation account](#).
- To enable from the Azure portal, see [Enable Change Tracking and Inventory from the Azure portal](#).
- To enable from a runbook, see [Enable Change Tracking and Inventory from a runbook](#).
- To enable from an Azure VM, see [Enable Change Tracking and Inventory from an Azure VM](#).

Supported regions for linked Log Analytics workspace

4/2/2021 • 3 minutes to read • [Edit Online](#)

In Azure Automation, you can enable the Update Management, Change Tracking and Inventory, and Start/Stop VMs during off-hours features for your servers and virtual machines. These features have a dependency on a Log Analytics workspace, and therefore require linking the workspace with an Automation account. However, only certain regions are supported to link them together. In general, the mapping is *not* applicable if you plan to link an Automation account to a workspace that won't have these features enabled.

The mappings discussed here apply only to linking the Log Analytics Workspace to an Automation account. They do not apply to the virtual machines (VMs) that are connected to the workspace that's linked to the Automation Account. VMs aren't limited to the regions supported by a given Log Analytics workspace. They can be in any region. Keep in mind that having the VMs in a different region may affect state, local, and country regulatory requirements, or your company's compliance requirements. Having VMs in a different region could also introduce data bandwidth charges.

Before connecting VMs to a workspace in a different region, you should review the requirements and potential costs to confirm and understand the legal and cost implications.

This article provides the supported mappings in order to successfully enable and use these features in your Automation account.

For more information, see [Log Analytics workspace and Automation account](#).

Supported mappings

NOTE

As shown in following table, only one mapping can exist between Log Analytics and Azure Automation.

The following table shows the supported mappings:

LOG ANALYTICS WORKSPACE REGION	AZURE AUTOMATION REGION
US	
EastUS ¹	EastUS2
EastUS ²	EastUS
WestUS	WestUS
WestUS2	WestUS2
NorthCentralUS	NorthCentralUS
CentralUS	CentralUS

LOG ANALYTICS WORKSPACE REGION	AZURE AUTOMATION REGION
SouthCentralUS	SouthCentralUS
WestCentralUS	WestCentralUS
Brazil	
BrazilSouth	BrazilSouth
Canada	
CanadaCentral	CanadaCentral
China	
ChinaEast2 ³	ChinaEast2
Asia Pacific	
EastAsia	EastAsia
SoutheastAsia	SoutheastAsia
India	
CentralIndia	CentralIndia
Japan	
JapanEast	JapanEast
Australia	
AustraliaEast	AustraliaEast
AustraliaSoutheast	AustraliaSoutheast
Korea	
KoreaCentral	KoreaCentral
Norway	
NorwayEast	NorwayEast
Europe	
NorthEurope	NorthEurope
WestEurope	WestEurope

LOG ANALYTICS WORKSPACE REGION	AZURE AUTOMATION REGION
France	
FranceCentral	FranceCentral
United Kingdom	
UKSouth	UKSouth
Switzerland	
SwitzerlandNorth	SwitzerlandNorth
United Arab Emirates	
UAENorth	UAENorth
US Gov	
USGovVirginia	USGovVirginia
USGovArizona ³	USGovArizona

¹ EastUS mapping for Log Analytics workspaces to Automation accounts isn't an exact region-to-region mapping, but is the correct mapping.

² EastUS2 mapping for Log Analytics workspaces to Automation accounts isn't an exact region-to-region mapping, but is the correct mapping.

³ In this region, only Update Management is supported, and other features like Change Tracking and Inventory are not available at this time.

Unlink a workspace

If you decide that you no longer want to integrate your Automation account with a Log Analytics workspace, you can unlink your account directly from the Azure portal. Before proceeding, you first need to [remove](#) Update Management, Change Tracking and Inventory, and Start/Stop VMs during off-hours if you are using them. If you don't remove them, you can't complete the unlinking operation.

With the features removed, you can follow the steps below to unlink your Automation account.

NOTE

Some features, including earlier versions of the Azure SQL monitoring solution, might have created Automation assets that need to be removed prior to unlinking the workspace.

- From the Azure portal, open your Automation account. On the Automation account page, select **Linked workspace** under **Related Resources**.
- On the Unlink workspace page, select **Unlink workspace**. You receive a prompt verifying if you want to continue.
- While Azure Automation is unlinking the account from your Log Analytics workspace, you can track the

progress under **Notifications** from the menu.

4. If you used Update Management, optionally you might want to remove the following items that are no longer needed:
 - Update schedules: Each has a name that matches an update deployment that you created.
 - Hybrid worker groups created for the feature: Each has a name similar to
`machine1.contoso.com_9ceb8108-26c9-4051-b6b3-227600d715c8`.
5. If you used Start/Stop VMs during off-hours, optionally you can remove the following items that are no longer needed:
 - Start and stop VM runbook schedules
 - Start and stop VM runbooks
 - Variables

Alternatively, you can unlink your workspace from your Automation account within the workspace.

1. In the workspace, select **Automation Account** under **Related Resources**.
2. On the Automation Account page, select **Unlink account**.

Next steps

- Learn about Update Management in [Update Management overview](#).
- Learn about Change Tracking and Inventory in [Change Tracking and Inventory overview](#).
- Learn about Start/Stop VMs during off-hours in [Start/Stop VMs during off-hours overview](#).

Enable Change Tracking and Inventory from Azure portal

3/5/2021 • 2 minutes to read • [Edit Online](#)

This article describes how you can enable [Change Tracking and Inventory](#) for one or more Azure VMs in the Azure portal. To enable Azure VMs at scale, you must enable an existing VM using Change Tracking and Inventory.

The number of resource groups that you can use for managing your VMs is limited by the [Resource Manager deployment limits](#). Resource Manager deployments are limited to five resource groups per deployment. Two of these resource groups are reserved to configure the Log Analytics workspace, Automation account, and related resources. This leaves you with three resource groups to select for management by Change Tracking and Inventory. This limit only applies to simultaneous setup, not the number of resource groups that can be managed by an Automation feature.

NOTE

When enabling Change Tracking and Inventory, only certain regions are supported for linking a Log Analytics workspace and an Automation Account. For a list of the supported mapping pairs, see [Region mapping for Automation Account and Log Analytics workspace](#).

Prerequisites

- Azure subscription. If you don't have one yet, you can [activate your MSDN subscriber benefits](#) or sign up for a [free account](#).
- [Automation account](#) to manage machines.
- A [virtual machine](#).

Sign in to Azure

Sign in to Azure at <https://portal.azure.com>.

Enable Change Tracking and Inventory

1. In the Azure portal, navigate to [Virtual machines](#).
2. Use the checkboxes to choose the VMs to add to Change Tracking and Inventory. You can add machines for up to three different resource groups at a time. Azure VMs can exist in any region, no matter the location of your Automation account.

The screenshot shows the Azure Virtual Machines dashboard under the 'Virtual machines' section. At the top, there are filter controls for 'Filter by name...', 'All resource groups', 'All types', 'All locations', and 'All tags'. A note at the top says: 'Try the new virtual machine resource browser! This experience is faster and has improved sorting and filtering capabilities. Please note that the new experience will not support all features available in the classic experience.' To the right, there's a sidebar with 'Change Tracking', 'Inventory', and 'Update Management' options.

Name	Type	Status	Resource group	Location	Source
CMPRI01	Virtual machine	Stopped (deallocated)	maic-rg	East US 2	Marketplace
DC01	Virtual machine	Running	MAIC-RG	East US 2	Marketplace
DC02	Virtual machine	Running	MAIC-RG	East US 2	Marketplace
DC03	Virtual machine	Running	MAIC-RG	East US 2	Marketplace
OMMS01	Virtual machine	Stopped (deallocated)	MAIC-RG	East US 2	Marketplace
OMMS02	Virtual machine	Stopped (deallocated)	MAIC-RG	East US 2	Marketplace
OMSQLServer01	Virtual machine	Stopped (deallocated)	maic-rg	East US 2	Marketplace
ProdWUS-DC01	Virtual machine	Stopped (deallocated)	ProdWUS	West US	Marketplace
slessvr01	Virtual machine	Stopped (deallocated)	MAIC-RG	East US 2	Marketplace
slessvr01	Virtual machine	Stopped (deallocated)	Prod1	East US	Marketplace
SVR01	Virtual machine	Running	MAIC-RG	East US 2	Marketplace

TIP

Use the filter controls to select VMs from different subscriptions, locations, and resource groups. You can click the top checkbox to select all virtual machines in a list.

3. Select **Change tracking or Inventory** under Configuration Management.
4. The list of virtual machines is filtered to show only the virtual machines that are in the same subscription and location. If your virtual machines are in more than three resource groups, the first three resource groups are selected.
5. An existing Log Analytics workspace and Automation account are selected by default. If you want to use a different Log Analytics workspace and Automation account, click **CUSTOM** to select them from the Custom Configuration page. When you choose a Log Analytics workspace, a check is made to determine if it is linked with an Automation account. If a linked Automation account is found, you see the following screen. When done, click **OK**.

The screenshot shows two overlapping dialogs. The left dialog is titled 'Enable Change Tracking' and contains sections for 'Change Tracking' (with a note about consistent control and compliance), 'Subscription' (set to 'Contoso'), 'Location' (set to 'East US 2'), and 'Configuration' (radio buttons for 'AUTO' and 'CUSTOM'). It also lists 'Log Analytics workspace' and 'Automation account' details. The right dialog is titled 'Custom Configuration' and lists 'Log Analytics workspace' (set to 'Contoso'), 'Location' (set to 'East US'), and 'Workspace' (set to 'Select a workspace'). Below these are summary counts for 'Ready to enable', 'Already enabled', and 'Cannot enable' VMs, and a table of VMs with their status (e.g., 'Ready to enable').

6. If the workspace selected isn't linked to an Automation account, you see the following screen. Select an Automation account and click **OK** when finished.

Custom Configuration

X

Log Analytics workspace

Subscription

Contoso

Location ⓘ

East US

Workspace

DefaultWorkspace38917



The workspace is not yet linked to an Automation account.

Select the account to link below.

Note: at this time linked account for this workspace must be from **East US 2**.

Automation account

Subscription

[dropdown menu]

Location ⓘ

East US 2

Account

Select an account

OK

Cancel

7. Deselect the checkbox next to any virtual machine that you don't want to enable. VMs that can't be enabled are already deselected.

8. Click **Enable** to enable the feature you've selected. The setup takes up to 15 minutes to complete.

Next steps

- For details of working with the feature, see [Manage Change Tracking](#) and [Manage Inventory](#).
- To troubleshoot general problems with the feature, see [Troubleshoot Change Tracking and Inventory issues](#).

Enable Change Tracking and Inventory from an Azure VM

3/5/2021 • 2 minutes to read • [Edit Online](#)

This article describes how you can use an Azure VM to enable [Change Tracking and Inventory](#) on other machines. To enable Azure VMs at scale, you must enable an existing VM using Change Tracking and Inventory.

NOTE

When enabling Change Tracking and Inventory, only certain regions are supported for linking a Log Analytics workspace and an Automation account. For a list of the supported mapping pairs, see [Region mapping for Automation account and Log Analytics workspace](#).

Prerequisites

- Azure subscription. If you don't have one yet, you can [activate your MSDN subscriber benefits](#) or sign up for a [free account](#).
- [Automation account](#) to manage machines.
- A [virtual machine](#).

Sign in to Azure

Sign in to the Azure portal at <https://portal.azure.com>.

Enable Change Tracking and Inventory

1. In the [Azure portal](#), select **Virtual machines** or search for and select **Virtual machines** from the Home page.
2. Select the VM for which you want to enable Change Tracking and Inventory. VMs can exist in any region, no matter the location of your Automation account.
3. On the VM page, select either **Inventory** or **Change tracking** under Configuration Management.
4. You must have the `Microsoft.OperationalInsights/workspaces/read` permission to determine if the VM is enabled for a workspace. To learn about additional permissions that are required, see [Feature setup permissions](#). To learn how to enable multiple machines at once, see [Enable Change Tracking and Inventory from an Automation account](#).
5. Choose the Log Analytics workspace and Automation account, and click **Enable** to enable Change Tracking and Inventory for the VM. The setup takes up to 15 minutes to complete.

Next steps

- For details of working with the feature, see [Manage Change Tracking](#) and [Manage Inventory](#).
- To troubleshoot general problems with the feature, see [Troubleshoot Change Tracking and Inventory issues](#).

Enable Change Tracking and Inventory from an Automation account

8/24/2021 • 3 minutes to read • [Edit Online](#)

This article describes how you can use your Automation account to enable [Change Tracking and Inventory](#) for VMs in your environment. To enable Azure VMs at scale, you must enable an existing VM using Change Tracking and Inventory.

NOTE

When enabling Change Tracking and Inventory, only certain regions are supported for linking a Log Analytics workspace and an Automation account. For a list of the supported mapping pairs, see [Region mapping for Automation account and Log Analytics workspace](#).

Prerequisites

- Azure subscription. If you don't have one yet, you can [activate your MSDN subscriber benefits](#) or sign up for a [free account](#).
- [Automation account](#) to manage machines.
- A [virtual machine](#).

Sign in to Azure

Sign in to Azure at <https://portal.azure.com>.

Enable Change Tracking and Inventory

1. Navigate to your Automation account and select either **Inventory** or **Change tracking** under **Configuration Management**.
2. Choose the Log Analytics workspace and Automation account and click **Enable** to enable Change Tracking and Inventory. The setup takes up to 15 minutes to complete.

Dashboard > Automation Accounts > Prod1-AA-Sec

Prod1-AA-Sec | Change tracking

Automation Account

Search (Ctrl+ /) <<

- Overview
- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems

Configuration Management

- Inventory
- Change tracking
- State configuration (DSC)

Update management

- Update management

Process Automation

- Runbooks
- Jobs
- Runbooks gallery

Change Tracking

Enable consistent control and compliance of your VMs with Change Tracking and Inventory.

This service is included with Azure virtual machines and Azure Arc machines. You only pay for logs stored in Log Analytics.

This service requires a Log Analytics workspace and this Automation account.

Log Analytics workspace location ⓘ

East US

Log Analytics workspace subscription ⓘ

Contoso

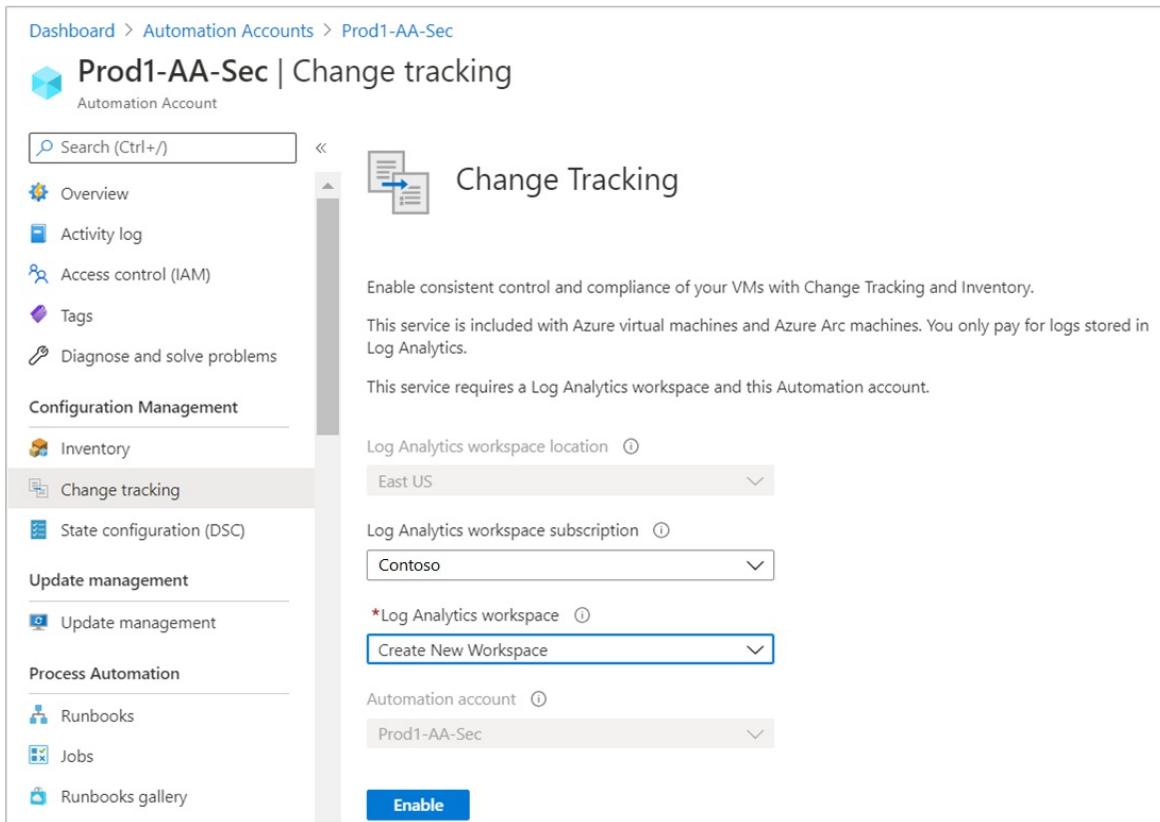
*Log Analytics workspace ⓘ

Create New Workspace

Automation account ⓘ

Prod1-AA-Sec

Enable



Enable Azure VMs

1. From your Automation account, select **Inventory** or **Change tracking** under Configuration Management.
2. Click **+ Add Azure VMs** and select one or more VMs from the list. Virtual machines that can't be enabled are grayed out and unable to be selected. Azure VMs can exist in any region no matter the location of your Automation account.
3. Click **Enable** to add the selected VMs to the computer group saved search for the feature. For more information, see [Limit Change Tracking and Inventory deployment scope](#).

Enable Change Tracking

The screenshot shows the 'Change Tracking' configuration page. At the top, there's a section titled 'Change Tracking' with a subtitle 'Enable consistent control and compliance of these virtual machines with Change Tracking.' Below this, there are two dropdown menus: 'Subscription' set to 'Contoso (virtual machines: 13)' and 'Location' set to 'East US 2 (virtual machines: 10)'. A configuration note says 'AUTO: Auto-configure Log Analytics workspace and Automation account based on VMs subscription and location'. The summary shows 'Ready to enable' (3), 'Already enabled' (0), and 'Cannot enable' (7). The main table lists four VMs: dc01 (Ready to enable), dc02 (Cannot enable), dc03 (Ready to enable), and omms01 (Cannot enable). At the bottom, there are 'Enable' and 'Cancel' buttons, and a note 'Number of virtual machines to enable Change Tracking: 3'.

Name	Status	Reason
dc01	Ready to enable	
dc02	Cannot enable	VM reports to workspace: 'infralabdefaultworkspace'
dc03	Ready to enable	
omms01	Cannot enable	VM reports to a workspace that is not available in this location

Enable non-Azure VMs

Machines not in Azure need to be added manually. We recommend installing the Log Analytics agent for Windows or Linux by first connecting your machine to [Azure Arc-enabled servers](#), and then using Azure Policy to assign the [Deploy Log Analytics agent to Linux or Windows Azure Arc machines](#) built-in policy. If you also plan to monitor the machines with Azure Monitor for VMs, instead use the [Enable Azure Monitor for VMs](#) initiative.

- From your Automation account select **Inventory** or **Change tracking** under **Configuration Management**.
- Click **Add non-Azure machine**. This action opens up a new browser window with [instructions to install and configure the Log Analytics agent for Windows](#) so that the machine can begin reporting Change Tracking and Inventory operations. If you're enabling a machine that's currently managed by Operations Manager, a new agent isn't required and the workspace information is entered into the existing agent.

Enable machines in the workspace

Manually installed machines or machines already reporting to your workspace must be added to Azure Automation for Change Tracking and Inventory to be enabled.

- From your Automation account, select **Inventory** or **Change tracking** under **Configuration Management**.
- Select **Manage machines**. The **Manage machines** option might be grayed out if you previously chose

the option **Enable on all available and future machines**

Manage Machines

Change Tracking and Inventory

These machines are reporting to the Log Analytics workspace 'maic-la', but they do not have 'Change Tracking and Inventory' enabled on them. It can take up to 15 minutes before data becomes available for machines that you enable with this feature. [Learn more](#)

Enable on all available machines
 Enable on all available and future machines
 Enable on selected machines

i For non-Azure machines that you enable with 'Change Tracking and Inventory' there may be additional monthly charge. [Learn more about pricing](#)

AVAILABLE MACHINES (2)	
DC02.Corp.Swanson.Local Azure: MAIC-RG/DC02	add
DC03 Azure: MAIC-RG/DC03	add

SELECTED MACHINES (0)

None

[add all from page](#)

Enable **Cancel**

3. To enable Change Tracking and Inventory for all available machines, select **Enable on all available machines** on the **Manage Machines** page. This action disables the control to add machines individually and adds all of the machines reporting to the workspace to the computer group saved search query. When selected, this action disables the **Manage Machines** option.
4. To enable the feature for all available machines and future machines, select **Enable on all available and future machines**. This option deletes the saved search and scope configuration from the workspace and opens the feature for all Azure and non-Azure machines that are reporting to the workspace. When selected, this action disables the **Manage Machines** option permanently, as there's no scope configuration left.

NOTE

Because this option deletes the saved search and scope configuration within Log Analytics, it's important to remove any deletion locks on the Log Analytics Workspace before you select this option. If you don't, the option will fail to remove the configurations and you must remove them manually.

5. If necessary, you can add the scope configuration back by re-adding the initial saved search. For more information, see [Limit Change Tracking and Inventory deployment scope](#).
6. To enable the feature for one or more machines, select **Enable on selected machines** and click **Add** next to each machine to enable for the feature. This task adds the selected machine names to the computer group saved search query for the feature.

Next steps

- For details of working with the feature, see [Manage Change Tracking](#) and [Manage Inventory](#).
- To troubleshoot general problems with the feature, see [Troubleshoot Change Tracking and Inventory issues](#).

Enable Change Tracking and Inventory from a runbook

3/5/2021 • 4 minutes to read • [Edit Online](#)

This article describes how you can use a runbook to enable [Change Tracking and Inventory](#) for VMs in your environment. To enable Azure VMs at scale, you must enable an existing VM using Change Tracking and Inventory.

NOTE

When enabling Change Tracking and Inventory, only certain regions are supported for linking a Log Analytics workspace and an Automation account. For a list of the supported mapping pairs, see [Region mapping for Automation account and Log Analytics workspace](#).

This method uses two runbooks:

- **Enable-MultipleSolution** - The primary runbook that prompts for configuration information, queries the specified VM and performs other validation checks, and then invokes the **Enable-AutomationSolution** runbook to configure Change Tracking and Inventory for each VM within the specified resource group.
- **Enable-AutomationSolution** - Enables Change Tracking and Inventory for one or more VMs specified in the target resource group. It verifies prerequisites are met, verifies the Log Analytics VM extension is installed and installs if not found, and adds the VMs to the scope configuration in the specified Log Analytics workspace linked to the Automation account.

Prerequisites

- Azure subscription. If you don't have one yet, you can [activate your MSDN subscriber benefits](#) or sign up for a [free account](#).
- [Automation account](#) to manage machines.
- [Log Analytics workspace](#)
- A [virtual machine](#).
- Two Automation assets, which are used by the **Enable-AutomationSolution** runbook. This runbook, if it doesn't already exist in your Automation account, is automatically imported by the **Enable-MultipleSolution** runbook during its first run.
 - *LASolutionSubscriptionId*: Subscription ID of where the Log Analytics workspace is located.
 - *LASolutionWorkspaceId*: Workspace ID of the Log Analytics workspace linked to your Automation account.

These variables are used to configure the workspace of the onboarded VM. If these are not specified, the script first searches for any VM onboarded to Change Tracking and Inventory in its subscription, followed by the subscription the Automation account is in, followed by all other subscriptions your user account has access to. If not properly configured, this may result in your machines getting onboarded to some random Log Analytics workspace.

Sign in to Azure

Sign in to the [Azure portal](#).

Enable Change Tracking and Inventory

1. In the Azure portal, navigate to **Automation Accounts**. On the **Automation Accounts** page, select your account from the list.
2. In your Automation account, select **Inventory** or **Change Tracking** under **Configuration Management**.
3. Select the Log Analytics workspace, then click **Enable**. While Inventory or Change Tracking is being enabled, a banner is shown.

The screenshot shows the 'Change tracking' configuration page for the 'Prod1-AA-Sec' Automation Account. The left sidebar lists various management options like Overview, Activity log, and Configuration Management. Under Configuration Management, 'Change tracking' is selected. The main pane displays a 'Change Tracking' section with a brief description: 'Enable consistent control and compliance of your VMs with Change Tracking and Inventory. This service is included with Azure virtual machines and Azure Arc machines. You only pay for logs stored in Log Analytics.' It also notes that 'This service requires a Log Analytics workspace and this Automation account.' Below this, there are dropdown menus for 'Log Analytics workspace location' (set to 'East US'), 'Log Analytics workspace subscription' (set to 'Contoso'), and 'Log Analytics workspace' (with a dropdown menu showing 'Create New Workspace'). At the bottom is an 'Enable' button.

Install and update modules

It's required to update to the latest Azure modules and import the `Az.OperationalInsights` module to successfully enable Update Management for your VMs with the runbook.

1. In your Automation account, select **Modules** under **Shared Resources**.
2. Select **Update Azure Modules** to update the Azure modules to the latest version.
3. Click **Yes** to update all existing Azure modules to the latest version.

Name	Last modified	Status
AuditPolicyDsc	9/16/2020, 4:15 AM	Available
Az.Accounts	6/19/2020, 4:07 PM	Available
Az.Automation	3/17/2020, 9:40 AM	Available
Az.Compute	6/19/2020, 4:09 PM	Available
Azure	11/7/2019, 4:13 PM	Available
Azure.Storage	1/31/2020, 3:14 PM	Available

4. Return to **Modules** under **Shared Resources**.
5. Select **Browse gallery** to open the module gallery.
6. Search for **Az.OperationalInsights** and import this module into your Automation account.

Az.OperationalInsights
PowerShell Module

Import

Microsoft Azure PowerShell - Operational Insights service cmdlets for Azure Resource Manager in Windows PowerShell and PowerShell Core.
Tags: Azure ResourceManager ARM Operation

Created by: azure-sdk
Tags: Azure ResourceManager ARM OperationalInsights PSModule PSEdition_Core PSEdition_Desktop
Dependencies: Az.Accounts (≥ 1.9.1)
View Source Project

Version: 2.3.0
1,650,692,541 downloads
Last updated: 7/16/2020

Learn more
[View in PowerShell Gallery](#)
[Documentation](#)
[Licensing information](#)

Content

Type **Name**

Cmdlet New-AzOperationalInsightsAzureActivityLogDataSource

Select Azure VM to manage

With Change Tracking and Inventory enabled, you can add an Azure VM for management by the feature.

1. From your Automation account, select **Change tracking or Inventory** under **Configuration Management**.
2. Click **Add Azure VMs** to add your VM.
3. Choose your VM from the list and click **Enable**. This action enables Change Tracking and Inventory for the VM.

Enable Change Tracking

 **Change Tracking**
Enable consistent control and compliance of these virtual machines with Change Tracking.

Subscription	Location
Contoso (virtual machines: 13)	East US 2 (virtual machines: 10)

Configuration (used when enabling new VMs) [?](#)
 AUTO: Auto-configure Log Analytics workspace and Automation account based on VMs subscription and location CUSTOM:

Log Analytics workspace: maic-la
Automation account: MAIC-AA-Pri

Summary
Ready to enable [?](#) Already enabled [?](#) Cannot enable [?](#)

Name	Change Tracking	Details
dc01	Ready to enable	
dc02	Cannot enable	VM is stopped. Start the VM
dc03	Cannot enable	VM is stopped. Start the VM
anmee01	Cannot enable	VM is stopped. Start the VM

Enable **Cancel** Number of virtual machines to enable Change Tracking: 1

NOTE

If you try to enable another feature before setup of Change Tracking and Inventory has completed, you receive this message:

Installation of another solution is in progress on this or a different virtual machine. When that installation completes the Enable button is enabled, and you can request installation of the solution on this virtual machine.

Import a runbook to enable Change Tracking and Inventory

1. In your Automation account, select **Runbooks** under Process Automation.
2. Select **Browse gallery**.
3. Search for **update and change tracking**.
4. Select the runbook and click **Import** on the **View Source** page.
5. Click **OK** to import the runbook into the Automation account.

The screenshot shows the Azure Automation Runbook Gallery interface. On the left, there's a search bar with 'update and change' typed in. A card for a runbook titled 'Onboard Azure VMs to update and change track' is displayed. This card includes a 'View Source' button and an 'Import' button. Below the card, there's a brief description: 'This sample automation runbook onboards Azure VMs to be onboarded to the solution.' It also lists tags: 'inventory, Windows Update, Windows Azure'. On the right, the runbook's source code is shown in a monospaced font. The code starts with a synopsis explaining its purpose: 'This sample automation runbook onboards Azure VMs for either the Update or Change Tracking (which includes Inventory). It requires an existing Azure VM to already be onboarded to the solution as it creates a new VM to the same Log Analytics workspace and Automation Account.' It also notes that the runbook needs to be run from the Automation account that has the 'Enable-AutomationSolution' runbook imported. The code continues with a detailed description of the runbook's functionality.

6. On the **Runbook** page, select the **Enable-MultipleSolution** runbook and then click **Edit**. On the textual editor, select **Publish**.
7. When prompted to confirm, click **Yes** to publish the runbook.

Start the runbook

You must have enabled Change Tracking and Inventory for an Azure VM to start this runbook. It requires an existing VM and resource group with the feature enabled in order to configure one or more VMs in the target resource group.

1. Open the **Enable-MultipleSolution** runbook.

The screenshot shows the Azure portal's runbook details page for 'Enable-MultipleSolution'. The top navigation bar includes 'Start', 'View', 'Edit', 'Link to schedule', 'Add webhook', 'Delete', and 'Export'. The main content area is divided into sections: 'Overview' (Resource group: ProdWUS, Account: ProdWUS-Pri-AA, Location: West US 2, Subscription: Contoso), 'Essentials' (Tags: Click here to add tags), 'Recent Jobs' (No jobs found), and 'Logs' (Recent logs: No logs found).

2. Click the start button and enter parameter values in the following fields:

- **VMNAME** - The name of an existing VM to add to Change Tracking and Inventory. Leave this field blank to add all VMs in the resource group.

- **VMRESOURCEGROUP** - The name of the resource group for the VMs to enable.
- **SUBSCRIPTIONID** - The subscription ID of the new VM to enable. Leave this field blank to use the subscription of the workspace. When you use a different subscription ID, add the Run As account for your Automation account as a contributor for the subscription.
- **ALREADYONBOARDEDVM** - The name of the VM that is already manually enabled for updates.
- **ALREADYONBOARDEDVMRESOURCEGROUP** - The name of the resource group to which the VM belongs.
- **SOLUTIONTYPE** - Enter **ChangeTracking**.

Start Runbook ↗ ✖

Enable-MultipleSolution

Parameters

VMNAME ⓘ
No value
Optional, String

VMRESOURCEGROUP * ⓘ
Finance ✓
Mandatory, String

SUBSCRIPTIONID ⓘ
No value
Optional, String

ALREADYONBOARDEDVM * ⓘ
DC01 ✓
Mandatory, String

ALREADYONBOARDEDVMRESOURCEGROUP * ⓘ
MAIC-RG ✓
Mandatory, String

SOLUTIONTYPE * ⓘ
ChangeTracking ✓
Mandatory, String

OK

3. Select **OK** to start the runbook job.
4. Monitor progress of the runbook job and any errors from the **Jobs** page.

Next steps

- To schedule a runbook, see [Manage schedules in Azure Automation](#).
- For details of working with the feature, see [Manage Change Tracking](#) and [Manage Inventory](#).

- To troubleshoot general problems with the feature, see [Troubleshoot Change Tracking and Inventory issues](#).

Manage Change Tracking and Inventory

6/22/2021 • 7 minutes to read • [Edit Online](#)

When you add a new file or registry key to track, Azure Automation enables it for [Change Tracking and Inventory](#). This article describes how to configure tracking, review tracking results, and handle alerts when changes are detected.

Before using the procedures in this article, ensure that you've enabled Change Tracking and Inventory on your VMs using one of these techniques:

- [Enable Change Tracking and Inventory from an Automation account](#)
- [Enable Change Tracking and Inventory by browsing the Azure portal](#)
- [Enable Change Tracking and Inventory from a runbook](#)
- [Enable Change Tracking and Inventory from an Azure VM](#)

Limit the scope for the deployment

Change Tracking and Inventory uses a scope configuration within the workspace to target the computers to receive changes. For more information, see [Limit Change Tracking and Inventory deployment scope](#).

Track files

You can use Change Tracking and Inventory to track changes to files and folders/directories. This section tells how to configure file tracking on Windows and on Linux.

Configure file tracking on Windows

Use the following steps to configure file tracking on Windows computers:

1. Sign in to the [Azure portal](#).
2. In the Azure portal, select **All services**. In the list of resources, type **Automation**. As you begin typing, the list filters suggestions based on your input. Select **Automation Accounts**.
3. In your list of Automation accounts, select the account you chose when you enabled Change Tracking and Inventory.
4. In your Automation account, select **Change tracking** under **Configuration Management**.
5. Select **Edit Settings** (the gear symbol).
6. On the Workspace Configuration page, select **Windows Files**, then click **+ Add** to add a new file to track.
7. On the Add Windows File for Change Tracking pane, enter the information for the file or folder to track and click **Save**. The following table defines the properties that you can use for the information.

PROPERTY	DESCRIPTION
Enabled	True if the setting is applied, and False otherwise.
Item Name	Friendly name of the file to be tracked.
Group	A group name for logically grouping files.
Enter Path	The path to check for the file, for example, <code>c:\temp*.txt</code> . You can also use environment variables, such as <code>%winDir%\System32*\.*</code> .

PROPERTY	DESCRIPTION
Path Type	The type of path. Possible values are File and Folder.
Recursion	True if recursion is used when looking for the item to be tracked, and False otherwise.
Upload file content	True to upload file content on tracked changes, and False otherwise.

If you plan on configuring monitoring of files and folders using wildcards, consider the following:

- Wildcards are required for tracking multiple files.
 - Wildcards can only be used in the last segment of a path, such as *C:\folder\file* or */etc/conf**
 - If an environment variable includes a path that is not valid, validation will succeed but the path will fail when inventory runs.
 - When setting the path, avoid general paths such as *c:*** which will result in too many folders being traversed.
8. Ensure that you specify True for **Upload file content**. This setting enables file content tracking for the indicated file path.

Configure file tracking on Linux

Use the following steps to configure file tracking on Linux computers:

- Select **Edit Settings** (the gear symbol).
- On the Workspace Configuration page, select **Linux Files**, then select **+ Add** to add a new file to track.
- On the **Add Linux File for Change Tracking** page, enter the information for the file or directory to track and then select **Save**. The following table defines the properties that you can use for the information.

PROPERTY	DESCRIPTION
Enabled	True if the setting is applied, and False otherwise.
Item Name	Friendly name of the file to be tracked.
Group	A group name for logically grouping files.
Enter Path	The path to check for the file, for example, <i>/etc/*.conf</i> .
Path Type	The type of path. Possible values are File and Directory.
Recursion	True if recursion is used when looking for the item to be tracked, and False otherwise.
Use Sudo	True to use sudo when checking for the item, and False otherwise.
Links	Setting that determines how to deal with symbolic links when traversing directories. Possible values are: Ignore - Ignores symbolic links and doesn't include the files/directories referenced. Follow - Follows the symbolic links during recursion and also includes the files/directories referenced. Manage - Follows the symbolic links and allows alteration of returned content. Note: The Manage option isn't recommended, as it doesn't support file content retrieval.

PROPERTY	DESCRIPTION
Upload file content	True to upload file content on tracked changes, and False otherwise.

4. Ensure that you specify **True** for **Upload file content**. This setting enables file content tracking for the indicated file path.

Add Linux File for Change Tracking X

Save Delete Discard

Enabled
True False

* Item Name
hosts ✓

Group
Custom

* Enter Path
/etc/hosts ✓

Path Type
File

Recursion
On Off

Use Sudo
On Off

Links
Follow

Upload file content for all settings
True False

Track file contents

File content tracking allows you to view the contents of a file before and after a tracked change. The feature saves the file contents to a [storage account](#) after each change occurs. Here are some rules to follow for tracking file contents:

- A standard storage account using the Resource Manager deployment model is required for storing file content.
- By default, storage accounts accept connections from clients on any network. If you have secured your storage account to allow only certain traffic, you need to modify your configuration rules to allow your Automation account to connect to it. See [Configure Azure Storage firewalls and virtual networks](#).
- Don't use premium and classic deployment model storage accounts. See [About Azure Storage accounts](#).
- You can connect the storage account to only one Automation account.
- Change Tracking and Inventory must be enabled in your Automation account.

Enable tracking for file content changes

Use the following steps to enable tracking for changes to file contents:

1. Select **Edit Settings** (the gear symbol).
2. Select **File Content** and then select **Link**. This selection opens the **Add Content Location for Change Tracking** page.

File Content Change Tracking allows you to view the file content of a changed file before and after the change. In order to provide this service, the file content data must be stored in a storage account after each change to that file is reported.

Turning on File Content Change Tracking and selecting a storage account will generate a shared access signature (SAS) URI that grants restricted write access to the selected storage. The SAS URI will grant the Change Tracking service access to write file content resources to the storage account for a specified period of time. During SAS URL generation, the container named "changetrackingblob" will be created in the selected storage account if it does not already exist. An access policy with write only permission will be added to the container, and the SAS URL will be created with the access policy. The SAS URL can be revoked anytime by deleting the access policy or by turning off File Content Change Tracking.

[View Privacy Statement](#)

Storage Account Name: [Link](#)

3. Select the subscription and storage account to use for storing the file contents.
4. If you want to enable file content tracking for all existing tracked files, select **On** for **Upload file content for all settings**. You can change this setting for each file path later.

Add Content Location for Chang... X

F Save Delete X Discard

Select Subscription ?
Microsoft Azure

Select Storage ?
changeblobexample

Upload file content for all settings ?
On Off

5. Change Tracking and Inventory shows storage account and Shared Access Signature (SAS) URLs when it enables file content change tracking. The signatures expire after 365 days, and you can recreate them by selecting **Regenerate**.

Storage Account Name ? changeblobexample [Unlink](#)

Primary Write Only SAS Url ? https://changeblobexample.blob.core.windows.net/changetrackingblob?sv=2015-04-05&sr=c&si=changetrackingpolicykey1_1530629144955&sig=MQj%2Buc0! [Regenerate](#)

Secondary Write Only SAS Url ? None [Regenerate](#)

View the contents of a tracked file

Once Change Tracking and Inventory detects a change for a tracked file, you can view the file contents on the Change Details pane.

RESOURCE NAME	CHANGE TYPE	MACHINE	TIME GENERATED	CATEGORY
/etc/hosts	Files	LinuxVM2	7/3/2018, 9:00 AM	Modified
rc-local	Daemons	LinuxVM2	7/3/2018, 8:40 AM	Removed
systemd-modules-load.kmod	Daemons	LinuxVM2	7/3/2018, 8:40 AM	Removed

1. On the **Change tracking** page from your Automation account, choose a file in the list of changes and then select **View File Content Changes** to see the contents of the file. The change details pane shows you the standard before and after file information for each property.

PROPERTY	VALUE BEFORE	VALUE AFTER
DateCreated	4/18/2018, 1:29:47 AM	7/3/2018, 8:56:20 AM
DateModified	4/16/2018, 2:02:20 PM	7/3/2018, 8:56:20 AM
FileContentCheck...	1523912540	1530633380
Size	221	237
▼ Unchanged properties		
SourceComput...	No Change	
FileSystemPath	/etc/hosts	No Change
SourceSystem	OpsManager	No Change
MG	00000000-0000-0000-0000-000000000002	No Change
TenantId	No Change	
VMUUID	No Change	

2. You're viewing the file contents in a side-by-side view. You can select **Inline** to see an inline view of the changes.

Track registry keys

Use the following steps to configure registry key tracking on Windows computers:

1. On the **Change tracking** page from your Automation account, select **Edit Settings** (the gear symbol).
2. On the Workspace Configuration page, select **Windows Registry**.
3. Select **+ Add** to add a new registry key to track.
4. On the **Add Windows Registry for Change Tracking** page, enter the information for the key to track and then select **Save**. The following table defines the properties that you can use for the information. When specifying a registry path, it must be the key and not a value.

PROPERTY	DESCRIPTION
Enabled	True if a setting is applied, and False otherwise.
Item Name	Friendly name of the registry key to track.
Group	Group name for logically grouping registry keys.
Windows Registry Key	<p>Key name with path, for example,</p> <pre>HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders\Common Startup</pre> <p>.</p>

Search logs for change records

You can do various searches against the Azure Monitor logs for change records. With the Change tracking page open, click **Log Analytics** to open the Logs page. The following table provides sample log searches for change records.

QUERY	DESCRIPTION
<pre>ConfigurationData where ConfigDataType == "WindowsServices" and SvcStartupType == "Auto" where SvcState == "Stopped" summarize arg_max(TimeGenerated, *) by SoftwareName, Computer</pre>	Shows the most recent inventory records for Windows services that were set to Auto but were reported as being Stopped. Results are limited to the most recent record for the specified software name and computer.
<pre>ConfigurationChange where ConfigChangeType == "Software" and ChangeCategory == "Removed" order by TimeGenerated desc</pre>	Shows change records for removed software.

Next steps

- For information about scope configurations, see [Limit Change Tracking and Inventory deployment scope](#).
- If you need to search logs stored in Azure Monitor Logs, see [Log searches in Azure Monitor Logs](#).
- If finished with deployments, see [Remove Change Tracking and Inventory](#).
- To delete your VMs from Change Tracking and Inventory, see [Remove VMs from Change Tracking and Inventory](#).
- To troubleshoot feature errors, see [Troubleshoot Change Tracking and Inventory issues](#).

Manage inventory collection from VMs

11/2/2020 • 5 minutes to read • [Edit Online](#)

You can enable inventory tracking for an Azure VM from the resource page of the machine. You can collect and view the following inventory information on your computers:

- Windows updates, Windows applications, services, files, and registry keys
- Linux software packages, daemons, and files

Azure Automation Change Tracking and Inventory provides a browser-based user interface for setting up and configuring inventory collection.

Before you begin

If you don't have an Azure subscription, [create a free account](#).

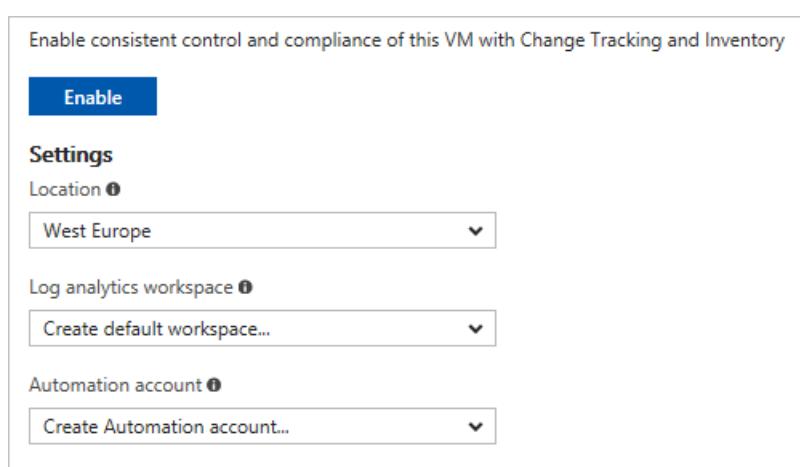
This article assumes that you have a VM to enable with Change Tracking and Inventory. If you don't have an Azure VM, you can [create a VM](#).

Sign in to the Azure portal

Sign in to the [Azure portal](#).

Enable inventory collection from the VM resource page

1. In the left pane of the Azure portal, select **Virtual machines**.
2. In the list of VMs, select a machine.
3. On the **Resource** menu, under **Operations**, select **Inventory**.
4. Select a Log Analytics workspace for storing your data logs. If no workspace is available to you for that region, you are prompted to create a default workspace and automation account.
5. To start enabling your computer, select **Enable**.



A status bar notifies you that the Change Tracking and Inventory feature is being enabled. This process can take up to 15 minutes. During this time, you can close the window, or you can keep it open and it notifies you when the feature is enabled. You can monitor the deployment status from the notifications pane.

The screenshot shows the Microsoft Endpoint Configuration Manager interface. At the top, there are navigation links: 'Manage multiple computers', 'Edit Settings', and 'Log Analytics'. Below this, a status bar indicates 'New software last 24 hrs' with a refresh icon and a count of '0'. To the right of the status bar are links for 'Learn more' and 'Provide feedback'. The main content area has tabs for 'Software', 'Files', 'Windows Registry', and 'Windows Services', with 'Software' being the active tab. A search bar says 'Search to filter items...'. A table below shows columns for 'NAME', 'VERSION', 'PUBLISHER', and 'LAST REFRESHED TIME'. A message 'No data' is displayed in the table.

When the deployment is complete, the status bar disappears. The system is still collecting inventory data, and the data might not be visible yet. A full collection of data can take 24 hours.

Configure your inventory settings

By default, software, Windows services, and Linux daemons are configured for collection. To collect Windows registry and file inventory, configure the inventory collection settings.

1. On the Inventory page, click **Edit Settings** at the top of the page.
2. To add a new collection setting, go to the setting category that you want to add by selecting the **Windows Registry**, **Windows Files**, or **Linux Files** tab.
3. Select the appropriate category and click **Add** at the top of the page.

The following sections provide information about each property that can be configured for the various categories.

Windows Registry

PROPERTY	DESCRIPTION
Enabled	Determines if the setting is applied
Item Name	Friendly name of the file to be tracked
Group	A group name for logically grouping files
Windows Registry Key	The path to check for the file For example: "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders\Common Startup"

Windows Files

PROPERTY	DESCRIPTION
Enabled	True if the setting is applied, and False otherwise.
Item Name	The friendly name of the file to be tracked.
Group	A group name for logically grouping files.
Enter Path	The path to check for the file, for example, c:\temp\myfile.txt.

Linux Files

PROPERTY	DESCRIPTION
Enabled	True if the setting is applied, and False otherwise.
Item Name	The friendly name of the file to be tracked.
Group	A group name for logically grouping files.
Enter Path	The path to check for the file, for example, /etc/*.*.conf.
Path Type	The type of item to be tracked. Values are File and Directory.
Recursion	True if recursion is used when looking for the item to be tracked, and False otherwise.
Use sudo	True if sudo is used when checking for the item, and False otherwise.
Links	Value indicating how symbolic links are dealt with when traversing directories. Possible values are: Ignore - Ignores symbolic links and does not include the files/directories referenced Follow - Follows the symbolic links during recursion and also includes the files/directories referenced Manage - Follows the symbolic links and allows alter the treatment of returned content

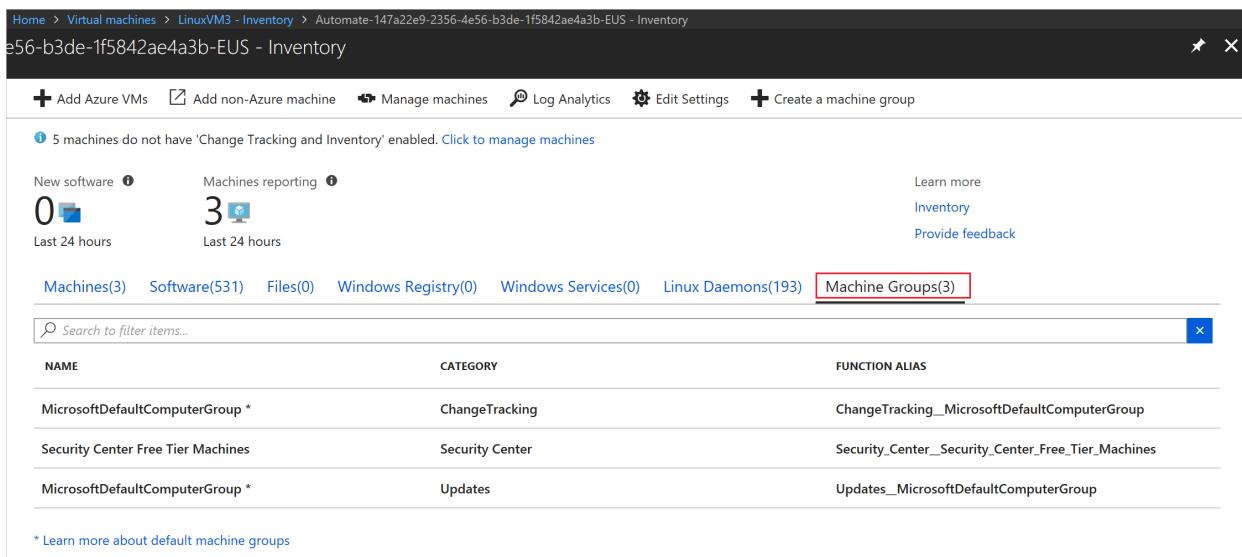
Manage machine groups

Inventory allows you to create and view machine groups in Azure Monitor logs. Machine groups are collections of machines defined by a query in Azure Monitor logs.

NOTE

This article was recently updated to use the term Azure Monitor logs instead of Log Analytics. Log data is still stored in a Log Analytics workspace and is still collected and analyzed by the same Log Analytics service. We are updating the terminology to better reflect the role of [logs in Azure Monitor](#). See [Azure Monitor terminology changes](#) for details.

To view your machine groups select the **Machine groups** tab on the Inventory page.



The screenshot shows the Azure Monitor Inventory interface. At the top, there's a navigation bar with links for Home, Virtual machines, LinuxVM3 - Inventory, and Automate-147a22e9-2356-4e56-b3de-1f5842ae4a3b-EUS - Inventory. Below the navigation is a toolbar with buttons for Add Azure VMs, Add non-Azure machine, Manage machines, Log Analytics, Edit Settings, and Create a machine group. A note indicates that 5 machines do not have 'Change Tracking and Inventory' enabled, with a link to manage them. The main area displays metrics for New software (0) and Machines reporting (3), both last 24 hours. Below this, a summary table shows counts for Machines(3), Software(531), Files(0), Windows Registry(0), Windows Services(0), Linux Daemons(193), and Machine Groups(3). The 'Machine Groups(3)' row is highlighted with a red box. At the bottom, there's a search bar and a note about default machine groups.

NAME	CATEGORY	FUNCTION ALIAS
MicrosoftDefaultComputerGroup *	ChangeTracking	ChangeTracking__MicrosoftDefaultComputerGroup
Security Center Free Tier Machines	Security Center	Security_Center__Security_Center_Free_Tier_Machines
MicrosoftDefaultComputerGroup *	Updates	Updates__MicrosoftDefaultComputerGroup

* Learn more about default machine groups

Selecting a machine group from the list opens the Machine groups page. This page shows details about the machine group. These details include the Azure Monitor log query that is used to define the group. At the bottom of the page, is a paged list of the machines that are part of that group.

The screenshot shows the 'Security_Center_Security_Center_Free_Tier_Machines' machine group configuration page. It includes fields for Group name (securitycenterfreetiermachines), Category (Security Center), and Function Alias (Security_Center__Security_Center_Free_Tier_Machines). The Definition section contains an OQL query: `Heartbeat | where SubscriptionId =~ "147a22e9-2356-4e56-b3de-1f5842ae4a3b" and ComputerEnvironment == "Azure" | distinct Computer | limit 200000 // Oql: Type=Heartbeat SubscriptionId=147a22e9-2356-4e56-b3de-1f5842ae4a3b and ComputerEnvironment=Azure | distinct Computer`. Below this, a table lists six machines: LinuxVM1, aks-agentpool-42602657-0, WinVM1, aks-agentpool-42602657-2, LinuxVM3, and aks-agentpool-42602657-1, all running Ubuntu 16.04 on Azure.

MACHINE	OPERATING SYSTEM	VERSION	PLATFORM
LinuxVM1	Linux	Ubuntu 16.04	Azure
aks-agentpool-42602657-0	Linux	Ubuntu 16.04	Azure
WinVM1	Windows	10.0	Azure
aks-agentpool-42602657-2	Linux	Ubuntu 16.04	Azure
LinuxVM3	Linux	Ubuntu 16.04	Azure
aks-agentpool-42602657-1	Linux	Ubuntu 16.04	Azure

Click **+ Clone** to clone the machine group. You must give the group a new name and alias for the group. The definition can be altered at this time. After changing the query, click **Validate query** to preview the machines that would be selected. When you are happy with the group, click **Create** to create the machine group.

If you want to create a new machine group, click **+ Create a machine group**. This button opens the **Create a machine group** page, where you can define your new group. Click **Create** to create the group.

Create a machine group

*** Group name** i
Computers_Needing_updates ✓

Category i
 Create new Use existing

Updates ▼

*** Function Alias** i
Computers_Needing_updates ✓

Definition i

```
Update
| where UpdateState == "Needed" and Optional == false
| summarize by Computer
```

Validate query

COMPUTER

WinVM1

Create

Disconnect your VM from management

To remove your VM from Change Tracking and Inventory management:

1. In the left pane of the Azure portal, select **Log Analytics**, and then select the workspace that you used when enabling your VM for Change Tracking and Inventory.
2. On the **Log Analytics** page, open the **Resource** menu.
3. Select **Virtual Machines** under **Workspace Data Sources**.
4. In the list, select the VM that you want to disconnect. The machine has a green check mark next to **This workspace** in the **OMS Connection** column.

NOTE

Operations Management Suite (OMS) is now referred to as Azure Monitor logs.

5. At the top of the next page, click **Disconnect**.
6. In the confirmation window, click **Yes** to disconnect the machine from management.

NOTE

Machines are still shown after you have unenrolled them because we report on all machines inventoried in the last 24 hours. After disconnecting the machine, you need to wait 24 hours before they are no longer listed.

Next steps

- For details of working with the feature, see [Manage Change Tracking and Inventory](#).
- To find out more about tracking software changes, see [Track software changes in your environment with Change Tracking](#).
- To troubleshoot general problems with the feature, see [Troubleshoot Change Tracking and Inventory issues](#).

Limit Change Tracking and Inventory deployment scope

5/28/2021 • 2 minutes to read • [Edit Online](#)

This article describes how to work with scope configurations when using the [Change Tracking and Inventory](#) feature to deploy changes to your VMs. For more information, see [Targeting monitoring solutions in Azure Monitor \(Preview\)](#).

About scope configurations

A scope configuration is a group of one or more saved searches (queries) used to limit the scope of Change Tracking and Inventory to specific computers. The scope configuration is used within the Log Analytics workspace to target the computers to enable. When you add a computer to changes from the feature, the computer is also added to a saved search in the workspace.

By default, Change Tracking and Inventory creates a computer group named **ChangeTracking__MicrosoftDefaultComputerGroup** depending on how you enabled machines:

- From the Automation account, you selected **+ Add Azure VMs**.
- From the Automation account, you selected **Manage machines**, and then you selected the option **Enable on all available machines** or you selected **Enable on selected machines**.

If one of the methods above is selected, this computer group is added to the **MicrosoftDefaultScopeConfig-ChangeTracking** scope configuration. You can also add one or more custom computer groups to this scope to match your management needs and control how specific computers are enabled for management with Change Tracking and Inventory.

To remove one or more machines from the **ChangeTracking__MicrosoftDefaultComputerGroup** to stop managing them with Change Tracking and Inventory, see [Remove VMs from Change Tracking and Inventory](#).

Set the scope limit

To limit the scope for your Change Tracking and Inventory deployment:

1. Sign in to the [Azure portal](#).
2. In the Azure portal, navigate to **Log Analytics workspaces**. Select your workspace from the list.
3. In your Log Analytics workspace, select **Scope Configurations (Preview)** from the left-hand menu.
4. Select the ellipsis to the right of the **MicrosoftDefaultScopeConfig-ChangeTracking** scope configuration, and select **Edit**.
5. In the editing pane, expand **Select Computer Groups**. The **Computer Groups** pane shows the saved searches that are added to the scope configuration. The saved search used by Update Management is:

NAME	CATEGORY	ALIAS
MicrosoftDefaultComputerGroup	ChangeTracking	ChangeTracking__MicrosoftDefaultComputerGroup

6. If you added a custom group, it is shown in the list. To deselect it, clear the checkbox to the left of the item.

To add a custom group to the scope, select it and then when you are finished with your changes, click **Select**.

7. On the **Edit scope configuration** page, click **OK** to save your changes.

Next steps

- To work with Change Tracking and Inventory, see [Manage Change Tracking and Inventory](#).
- To troubleshoot general feature issues, see [Troubleshoot Change Tracking and Inventory issues](#).

How to create alerts for Change Tracking and Inventory

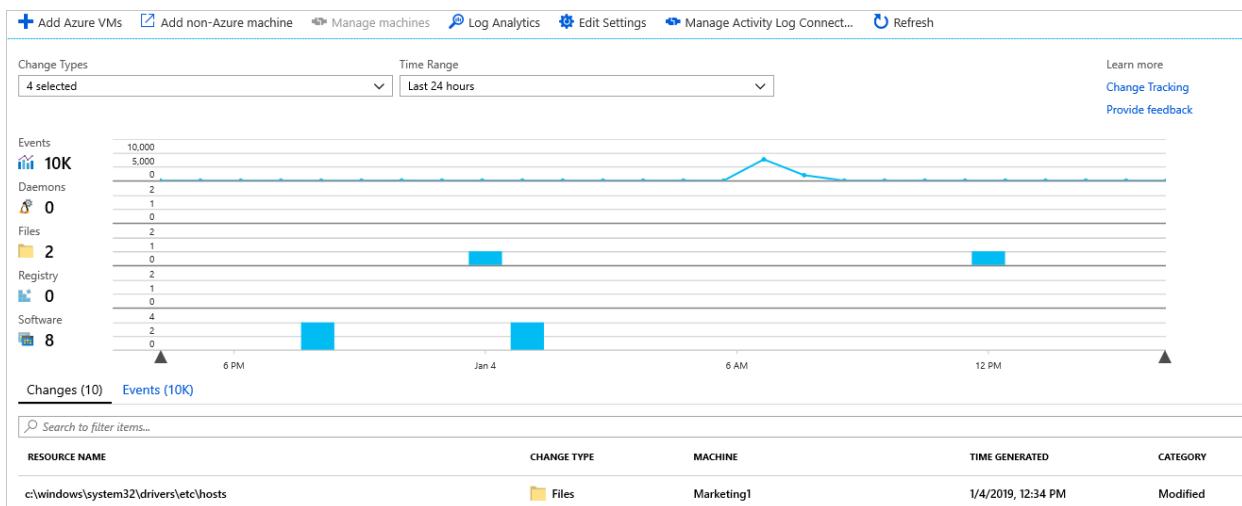
3/18/2021 • 3 minutes to read • [Edit Online](#)

Alerts in Azure proactively notify you of results from runbook jobs, service health issues, or other scenarios related to your Automation account. Azure Automation does not include pre-configured alert rules, but you can create your own based on data that it generates. This article provides guidance on creating alert rules based on changes identified by Change Tracking and Inventory.

If you're not familiar with Azure Monitor alerts, see [Overview of alerts in Microsoft Azure](#) before you start. To learn more about alerts that use log queries, see [Log alerts in Azure Monitor](#).

Create alert

The following example shows that the file `c:\windows\system32\drivers\etc\hosts` has been modified on a machine. This file is important because Windows uses it to resolve host names to IP addresses. This operation takes precedence over DNS, and might result in connectivity issues. It can also lead to redirection of traffic to malicious or otherwise dangerous websites.



Let's use this example to discuss the steps for creating alerts on a change.

1. On the **Change tracking** page from your Automation account, select **Log Analytics**.
2. In the Logs search, look for content changes to the **hosts** file with the query

```
ConfigurationChange | where FieldsChanged contains "FileContentChecksum" and FileSystemPath contains "hosts"
```

. This query looks for content changes for files with fully qualified path names containing the word `hosts`. You can also ask for a specific file by changing the path portion to its fully qualified form, for example, using `FileSystemPath == "c:\windows\system32\drivers\etc\hosts"`.
3. After the query returns its results, select **New alert rule** in the log search to open the **Alert creation** page. You can also navigate to this page through **Azure Monitor** in the Azure portal.
4. Check your query again and modify the alert logic. In this case, you want the alert to be triggered if there's even one change detected across all the machines in the environment.

*** Search query** ⓘ
ConfigurationChange | where FieldsChanged contains "FileContentChecksum" and FileSystemPath ✓ contains "hosts"

Query to be executed : ConfigurationChange | where FieldsChanged contains "FileContentChecksum" and FileSystemPath contains "hosts" ↵| count
For time window : 1/4/2019, 4:17:49 PM - 1/4/2019, 4:22:49 PM

Alert logic

Based on ⓘ Condition ⓘ * Threshold ⓘ

Number of results	Greater than or equal to	1
-------------------	--------------------------	---

Condition preview
Whenever the custom log search is greater than 1 count

Evaluated based on

* Period (in minutes) ⓘ * Frequency (in minutes) ⓘ

5	5
---	---

- After the alert logic is set, assign action groups to perform actions in response to triggering of the alert.
In this case, we're setting up emails to be sent and an IT Service Management (ITSM) ticket to be created.

Follow the steps below to set up alerts to let you know the status of an update deployment. If you are new to Azure alerts, see [Azure Alerts overview](#).

Configure action groups for your alerts

Once you have your alerts configured, you can set up an action group, which is a group of actions to use across multiple alerts. The actions can include email notifications, runbooks, webhooks, and much more. To learn more about action groups, see [Create and manage action groups](#).

- Select an alert and then select **Create New** under **Action Groups**.
- Enter a full name and a short name for the action group. Update Management uses the short name when sending notifications using the specified group.
- Under **Actions**, enter a name that specifies the action, for example, **Email Notification**.
- For **Action Type**, select the appropriate type, for example, **Email/SMS message/Push/Voice**.
- Select the pencil icon to edit the action details.
- Fill in the pane for your action type. For example, if using **Email/SMS message/Push/Voice** to send an email, enter an action name, select the **Email** checkbox, enter a valid email address, and then select **OK**.

Home > MAIC-AA-Pri > Manage actions >
StartStop_VM_Notification ⚙ ...
Edit action group

Save changes Delete action group

This is a summary of your action group. Please review to ensure the information is correct or

Basics

Subscription

Bellows College

Resource group

prod1

Action group name

StartStop_VM_Notification

Display name *

StStAlert

Notifications

Notification type	Name	Status
Email/SMS message/Push/Voice	EmailInt3m3rhm56gq4	Subscrib

Actions

Action type	Name

Email/SMS message/Push/Voice

Add or edit an Email/SMS/Push/Voice action

Email

Email *

SMS (Carrier charges may apply)

Country code

1

Phone number

Azure app Push Notifications

Azure account email

Voice

Country code

1

Phone number

Enable the common alert schema. [Learn more](#)

Yes No

OK

7. In the Add action group pane, select **OK**.

8. For an alert email, you can customize the email subject. Select **Customize actions** under **Create rule**, then select **Email subject**.

9. When you're finished, select **Create alert rule**.

Next steps

- Learn more about [alerts in Azure Monitor](#).
- Learn about [log queries](#) to retrieve and analyze data from a Log Analytics workspace.
- Manage [usage and costs with Azure Monitor Logs](#) describes how to control your costs by changing your data retention period, and how to analyze and alert on your data usage.

Remove Change Tracking and Inventory from Automation account

11/2/2020 • 2 minutes to read • [Edit Online](#)

After you enable management of your virtual machines using Azure Automation Change Tracking and Inventory, you may decide to stop using it and remove the configuration from the account and linked Log Analytics workspace. This article tells you how to completely remove Change Tracking and Inventory from the managed VMs, your Automation account, and Log Analytics workspace.

Sign into the Azure portal

Sign in to the [Azure portal](#).

Remove management of VMs

Before removing Change Tracking and Inventory, you need to first stop managing your VMs. See [Remove VMs from Change Tracking](#) to unenroll them from the feature.

Remove ChangeTracking solution

Before you can unlink the Automation account from the workspace, you need to follow these steps to completely remove Change Tracking and Inventory. You'll remove the **ChangeTracking** solution from the workspace.

1. In the Azure portal, select **All services**. In the list of resources, type **Log Analytics**. As you begin typing, the list filters suggestions based on your input. Select **Log Analytics**.
2. In your list of Log Analytics workspaces, select the workspace you chose when you enabled Change Tracking and Inventory.
3. On the left, select **Solutions**.
4. In the list of solutions, select **ChangeTracking(workspace name)**. On the **Overview** page for the solution, select **Delete**. When prompted to confirm, select **Yes**.

Unlink workspace from Automation account

1. In the Azure portal, select **Automation Accounts**.
2. Open your Automation account and select **Linked workspace** under **Related Resources** on the left.
3. On the **Unlink workspace** page, select **Unlink workspace** and respond to prompts.

The screenshot shows the Azure portal interface for an Automation Account named 'TestAzureAuto - Linked workspace'. On the left, there's a sidebar with 'Search (Ctrl+)' at the top, followed by 'Connections', 'Certificates', and 'Variables'. Under 'RELATED RESOURCES', 'Linked workspace' is selected, showing a list of three items: 'Event grid', 'Start/Stop VM', and another 'Linked workspace' item. Under 'ACCOUNT SETTINGS', there are 'Properties' and 'Keys'. The main content area is titled 'Linked workspace' and states: 'This Automation account is linked to the following Log Analytics workspace: workspace-e7e3582f-a695-46cb-bc8a-b63f08f57046-eus'. Below this, there's a section for 'Unlink workspace' with instructions and a list of solutions to remove. A note says 'After you remove these solutions you can click **Unlink workspace** above to complete the unlinking.' There's also a link to 'View schedules'.

While it attempts to unlink the Log Analytics workspace, you can track the progress under **Notifications** from the menu.

Alternatively, you can unlink your Log Analytics workspace from your Automation account from within the workspace:

1. In the Azure portal, select **Log Analytics**.
2. From the workspace, select **Automation Account** under **Related Resources**.
3. On the Automation Account page, select **Unlink account**.

While it attempts to unlink the Automation account, you can track the progress under **Notifications** from the menu.

Next steps

To re-enable this feature, see [Enable Change Tracking and Inventory from an Automation account](#), [Enable Change Tracking and Inventory by browsing the Azure portal](#), [Enable Change Tracking and Inventory from a runbook](#), or [Enable Change Tracking and Inventory from an Azure VM](#).

Remove VMs from Change Tracking and Inventory

5/28/2021 • 2 minutes to read • [Edit Online](#)

When you're finished tracking changes on the VMs in your environment, you can stop managing them with the [Change Tracking and Inventory](#) feature. To stop managing them, you will edit the saved search query `MicrosoftDefaultComputerGroup` in your Log Analytics workspace that is linked to your Automation account.

Sign into the Azure portal

Sign in to the [Azure portal](#).

To remove your VMs

1. In the Azure portal, launch **Cloud Shell** from the top navigation of the Azure portal. If you are unfamiliar with Azure Cloud Shell, see [Overview of Azure Cloud Shell](#).
2. Use the following command to identify the UUID of a machine that you want to remove from management.

```
az vm show -g MyResourceGroup -n MyVm -d
```
3. In the Azure portal, navigate to **Log Analytics workspaces**. Select your workspace from the list.
4. In your Log Analytics workspace, select **Computer Groups** from the left-hand menu.
5. From **Computer Groups** in the right-hand pane, the **Saved groups** tab is shown by default.
6. From the table, click the icon **Run query** to the right of the item `MicrosoftDefaultComputerGroup` with the **Legacy category** value `ChangeTracking`.
7. In the query editor, review the query and find the UUID for the VM. Remove the UUID for the VM and repeat the steps for any other VMs you want to remove.

NOTE

For added protection, before making edits be sure to make a copy of the query. Then you can restore it if a problem occurs.

If you want to start with the original query and re-add machines in support of a cleanup or maintenance activity, copy the following query:

```
Heartbeat  
| where Computer in~ ("") or VMUUID in~ ("")  
| distinct Computer
```

8. Save the saved search when you're finished editing it by selecting **Save > Save as function** from the top bar. When prompted, specify the following:
 - **Name:** `ChangeTracking__MicrosoftDefaultComputerGroup`
 - **Save as computer Group** is selected
 - **Legacy category:** `ChangeTracking`

NOTE

Machines are still shown after you have unenrolled them because we report on all machines assessed in the last 24 hours. After removing the machine, you need to wait 24 hours before they are no longer listed.

Next steps

To re-enable this feature, see [Enable Change Tracking and Inventory from an Automation account](#), [Enable Change Tracking and Inventory by browsing the Azure portal](#), [Enable Change Tracking and Inventory from a runbook](#), or [Enable Change Tracking and Inventory from an Azure VM](#).

Troubleshoot feature deployment issues

8/20/2021 • 6 minutes to read • [Edit Online](#)

You might receive error messages when you deploy the Azure Automation Update Management feature or the Change Tracking and Inventory feature on your VMs. This article describes the errors that might occur and how to resolve them.

Known issues

Scenario: Renaming a registered node requires unregister or register again

Issue

A node is registered to Azure Automation, and then the operating system computer name is changed. Reports from the node continue to appear with the original name.

Cause

Renaming registered nodes doesn't update the node name in Azure Automation.

Resolution

Unregister the node from Azure Automation State Configuration, and then register it again. Reports published to the service before that time will no longer be available.

Scenario: Re-signing certificates via HTTPS proxy isn't supported

Issue

When you connect through a proxy that terminates HTTPS traffic and then re-encrypts the traffic using a new certificate, the service doesn't allow the connection.

Cause

Azure Automation doesn't support re-signing certificates used to encrypt traffic.

Resolution

There's currently no workaround for this issue.

General errors

Scenario: Feature deployment fails with the message "The solution cannot be enabled"

Issue

You receive one of the following messages when you attempt to enable a feature on a VM:

The solution cannot be enabled due to missing permissions for the virtual machine or deployments

The solution cannot be enabled on this VM because the permission to read the workspace is missing

Cause

This error is caused by incorrect or missing permissions on the VM or workspace, or for the user.

Resolution

Ensure that you have correct [feature deployment permissions](#), and then try to deploy the feature again. If you receive the error message

The solution cannot be enabled on this VM because the permission to read the workspace is missing, see the following [troubleshooting information](#).

Scenario: Feature deployment fails with the message "Failed to configure automation account for diagnostic logging"

Issue

You receive the following message when you attempt to enable a feature on a VM:

```
Failed to configure automation account for diagnostic logging
```

Cause

This error can be caused if the pricing tier doesn't match the subscription's billing model. For more information, see [Monitoring usage and estimated costs in Azure Monitor](#).

Resolution

Create your Log Analytics workspace manually, and repeat the feature deployment process to select the workspace created.

Scenario: ComputerGroupQueryFormatError

Issue

This error code means that the saved search computer group query used to target the feature isn't formatted correctly.

Cause

You might have altered the query, or the system might have altered it.

Resolution

You can delete the query for the feature and then enable the feature again, which re-creates the query. The query can be found in your workspace under [Saved searches](#). The name of the query is [MicrosoftDefaultComputerGroup](#), and the category of the query is the name of the associated feature. If multiple features are enabled, the [MicrosoftDefaultComputerGroup](#) query shows multiple times under [Saved searches](#).

Scenario: PolicyViolation

Issue

This error code indicates that the deployment failed due to violation of one or more Azure Policy assignments.

Cause

An Azure Policy assignment is blocking the operation from completing.

Resolution

To successfully deploy the feature, you must consider altering the indicated policy definition. Because there are many different types of policy definitions that can be defined, the changes required depend on the policy definition that's violated. For example, if a policy definition is assigned to a resource group that denies permission to change the contents of some contained resources, you might choose one of these fixes:

- Remove the policy assignment altogether.
- Try to enable the feature for a different resource group.
- Retarget the policy assignment to a specific resource, for example, an Automation account.
- Revise the set of resources that the policy definition is configured to deny.

Check the notifications in the upper-right corner of the Azure portal, or go to the resource group that contains your Automation account and select [Deployments](#) under [Settings](#) to view the failed deployment. To learn more about Azure Policy, see [Overview of Azure Policy](#).

Scenario: Errors trying to unlink a workspace

Issue

You receive the following error message when you try to unlink a workspace:

The link cannot be updated or deleted because it is linked to Update Management and/or ChangeTracking Solutions.

Cause

This error occurs when you still have features active in your Log Analytics workspace that depend on your Automation account and Log Analytics workspace being linked.

Resolution

Remove the resources for the following features from your workspace if you're using them:

- Update Management
- Change Tracking and Inventory
- Start/Stop VMs during off-hours

After you remove the feature resources, you can unlink your workspace. It's important to clean up any existing artifacts from these features from your workspace and your Automation account:

- For Update Management, remove **Update Deployments (Schedules)** from your Automation account.
- For Start/Stop VMs during off-hours, remove any locks on feature components in your Automation account under **Settings > Locks**. For more information, see [Remove the feature](#).

Log Analytics for Windows extension failures

NOTE

As part of the ongoing transition from Microsoft Operations Management Suite to Azure Monitor, the Operations Management Suite Agent for Windows or Linux will be referred to as the Log Analytics agent for Windows and Log Analytics agent for Linux.

An installation of the Log Analytics agent for Windows extension can fail for a variety of reasons. The following section describes feature deployment issues that can cause failures during deployment of the Log Analytics agent for Windows extension.

NOTE

Log Analytics agent for Windows is the name used currently in Azure Automation for the Microsoft Monitoring Agent (MMA).

Scenario: An exception occurred during a WebClient request

The Log Analytics for Windows extension on the VM is unable to communicate with external resources and the deployment fails.

Issue

The following are examples of error messages that are returned:

Please verify the VM has a running VM agent, and can establish outbound connections to Azure storage.

'Manifest download error from
https://<endpoint>/<endpointId>/Microsoft.EnterpriseCloud.Monitoring_MicrosoftMonitoringAgent_australiaeast_manifest.xml. Error: UnknownError. An exception occurred during a WebClient request.'

Cause

Some potential causes of this error are:

- A proxy configured in the VM only allows specific ports.
- A firewall setting has blocked access to the required ports and addresses.

Resolution

Ensure that you have the proper ports and addresses open for communication. For a list of ports and addresses, see [Planning your network](#).

Scenario: Install failed because of transient environment issues

The installation of the Log Analytics for Windows extension failed during deployment because of another installation or action blocking the installation.

Issue

The following are examples of error messages that might be returned:

```
The Microsoft Monitoring Agent failed to install on this machine. Please try to uninstall and reinstall the extension. If the issue persists, please contact support.
```

```
'Install failed for plugin (name: Microsoft.EnterpriseCloud.Monitoring.MicrosoftMonitoringAgent, version 1.0.11081.4) with exception Command C:\Packages\Plugins\Microsoft.EnterpriseCloud.Monitoring.MicrosoftMonitoringAgent\1.0.11081.4\MMAExtensionInstall.exe of Microsoft.EnterpriseCloud.Monitoring.MicrosoftMonitoringAgent has exited with Exit code: 1618'
```

```
'Install failed for plugin (name: Microsoft.EnterpriseCloud.Monitoring.MicrosoftMonitoringAgent, version 1.0.11081.2) with exception Command C:\Packages\Plugins\Microsoft.EnterpriseCloud.Monitoring.MicrosoftMonitoringAgent\1.0.11081.2\MMAExtensionInstall.exe of Microsoft.EnterpriseCloud.Monitoring.MicrosoftMonitoringAgent has exited with Exit code: 1601'
```

Cause

Some potential causes of this error are:

- Another installation is in progress.
- The system is triggered to reboot during template deployment.

Resolution

This error is transient in nature. Retry the deployment to install the extension.

Scenario: Installation timeout

The installation of the Log Analytics for Windows extension didn't complete because of a timeout.

Issue

The following is an example of an error message that might be returned:

```
Install failed for plugin (name: Microsoft.EnterpriseCloud.Monitoring.MicrosoftMonitoringAgent, version 1.0.11081.4) with exception Command C:\Packages\Plugins\Microsoft.EnterpriseCloud.Monitoring.MicrosoftMonitoringAgent\1.0.11081.4\MMAExtensionInstall.exe of Microsoft.EnterpriseCloud.Monitoring.MicrosoftMonitoringAgent has exited with Exit code: 15614
```

Cause

This type of error occurs because the VM is under a heavy load during installation.

Resolution

Try to install the Log Analytics agent for Windows extension when the VM is under a lower load.

Next steps

If you don't see your problem here or you can't resolve your issue, try one of the following channels for additional support:

- Get answers from Azure experts through [Azure Forums](#).
- Connect with [@AzureSupport](#), the official Microsoft Azure account for improving customer experience. Azure Support connects the Azure community to answers, support, and experts.
- File an Azure support incident. Go to the [Azure support site](#), and select **Get Support**.

Troubleshoot Change Tracking and Inventory issues

3/5/2021 • 4 minutes to read • [Edit Online](#)

This article describes how to troubleshoot and resolve Azure Automation Change Tracking and Inventory issues. For general information about Change Tracking and Inventory, see [Change Tracking and Inventory overview](#).

General errors

Scenario: Machine is already registered to a different account

Issue

You receive the following error message:

```
Unable to Register Machine for Change Tracking, Registration Failed with Exception  
System.InvalidOperationException: {"Message":"Machine is already registered to a different account."}
```

Cause

The machine has already been deployed to another workspace for Change Tracking.

Resolution

1. Make sure that your machine is reporting to the correct workspace. For guidance on how to verify this, see [Verify agent connectivity to Azure Monitor](#). Also make sure that this workspace is linked to your Azure Automation account. To confirm, go to your Automation account and select **Linked workspace** under **Related Resources**.
2. Make sure that the machines show up in the Log Analytics workspace linked to your Automation account. Run the following query in the Log Analytics workspace.

```
Heartbeat  
| summarize by Computer, Solutions
```

If you don't see your machine in the query results, it hasn't checked in recently. There's probably a local configuration issue. You should reinstall the Log Analytics agent.

If your machine is listed in the query results, verify under the Solutions property that **changeTracking** is listed. This verifies it is registered with Change Tracking and Inventory. If it is not, check for scope configuration problems. The scope configuration determines which machines are configured for Change Tracking and Inventory. To configure the scope configuration for the target machine, see [Enable Change Tracking and Inventory from an Automation account](#).

In your workspace, run this query.

```
Operation  
| where OperationCategory == 'Data Collection Status'  
| sort by TimeGenerated desc
```

3. If you get a

`Data collection stopped due to daily limit of free data reached. Ingestion status = OverQuota` result, the quota defined on your workspace has been reached, which has stopped data from being saved. In your workspace, go to **Usage and estimated costs**. Either select a new **Pricing tier** that allows you to

use more data, or click on **Daily cap**, and remove the cap.

The screenshot shows the Microsoft Azure portal with the URL [https://logAnalyticsWorkspaces.azure.com/#/workspaces/MAIC-LA](#). The left sidebar shows navigation options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings, General, and Properties. The main content area is titled "MAIC-LA | Usage and estimated costs". It displays usage details, a daily cap section, and a data retention section. The "Daily cap" section is highlighted with a red box. It contains a chart titled "Usage Chart" showing billable data ingestion from Jan 17 to Jan 24, with a current value of 0MB. A switch labeled "ON" is shown, which is currently set to "OFF". A note says "Be sure to create an alert so you know if your workspace is capped." At the bottom right of the main content area is an "OK" button.

If your issue is still unresolved, follow the steps in [Deploy a Windows Hybrid Runbook Worker](#) to reinstall the Hybrid Worker for Windows. For Linux, follow the steps in [Deploy a Linux Hybrid Runbook Worker](#).

Windows

Scenario: Change Tracking and Inventory records aren't showing for Windows machines

Issue

You don't see any Change Tracking and Inventory results for Windows machines that have been enabled for the feature.

Cause

This error can have the following causes:

- The Azure Log Analytics agent for Windows isn't running.
- Communication back to the Automation account is being blocked.
- The Change Tracking and Inventory management packs aren't downloaded.
- The VM being enabled might have come from a cloned machine that wasn't prepared with System Preparation (sysprep) with the Log Analytics agent for Windows installed.

Resolution

On the Log Analytics agent machine, go to **C:\Program Files\Microsoft Monitoring Agent\Agent\Tools** and run the following commands:

```
net stop healthservice
StopTracing.cmd
StartTracing.cmd VER
net start healthservice
```

If you still need help, you can collect diagnostics information and contact support.

NOTE

The Log Analytics agent enables error tracing by default. To enable verbose error messages as in the preceding example, use the `VER` parameter. For information traces, use `INF` when you invoke `StartTracing.cmd`.

Log Analytics agent for Windows not running

Verify that the Log Analytics agent for Windows (**HealthService.exe**) is running on the machine.

Communication to Automation account blocked

Check Event Viewer on the machine, and look for any events that have the word `changetracking` in them.

To learn about addresses and ports that must be allowed for Change Tracking and Inventory to work, see [Network planning](#).

Management packs not downloaded

Verify that the following Change Tracking and Inventory management packs are installed locally:

- `Microsoft.IntelligencePacks.ChangeTrackingDirectAgent.*`
- `Microsoft.IntelligencePacks.InventoryChangeTracking.*`
- `Microsoft.IntelligencePacks.SingletonInventoryCollection.*`

VM from cloned machine that has not been sysprepped

If using a cloned image, sysprep the image first and then install the Log Analytics agent for Windows.

Linux

Scenario: No Change Tracking and Inventory results on Linux machines

Issue

You don't see any Change Tracking and Inventory results for Linux machines that are enabled for the feature.

Cause

Here are possible causes specific to this issue:

- The Log Analytics agent for Linux isn't running.
- The Log Analytics agent for Linux isn't configured correctly.
- There are file integrity monitoring (FIM) conflicts.

Resolution

Log Analytics agent for Linux not running

Verify that the daemon for the Log Analytics agent for Linux (**omsagent**) is running on your machine. Run the following query in the Log Analytics workspace that's linked to your Automation account.

```
Heartbeat  
| summarize by Computer, Solutions
```

If you don't see your machine in query results, it hasn't recently checked in. There's probably a local configuration issue and you should reinstall the agent. For information about installation and configuration, see [Collect log data with the Log Analytics agent](#).

If your machine shows up in the query results, verify the scope configuration. See [Targeting monitoring solutions in Azure Monitor](#).

For more troubleshooting of this issue, see [Issue: You are not seeing any Linux data](#).

Log Analytics agent for Linux not configured correctly

The Log Analytics agent for Linux might not be configured correctly for log and command-line output collection by using the OMS Log Collector tool. See [Change Tracking and Inventory overview](#).

FIM conflicts

Azure Security Center's FIM feature might be incorrectly validating the integrity of your Linux files. Verify that FIM is operational and correctly configured for Linux file monitoring. See [Change Tracking and Inventory overview](#).

Next steps

If you don't see your problem here or you can't resolve your issue, try one of the following channels for additional support:

- Get answers from Azure experts through [Azure Forums](#).
- Connect with [@AzureSupport](#), the official Microsoft Azure account for improving customer experience. Azure Support connects the Azure community to answers, support, and experts.
- File an Azure support incident. Go to the [Azure Support site](#), and select **Get Support**.

Start/Stop VMs during off-hours overview

5/25/2021 • 12 minutes to read • [Edit Online](#)

The Start/Stop VMs during off-hours feature start or stops enabled Azure VMs. It starts or stops machines on user-defined schedules, provides insights through Azure Monitor logs, and sends optional emails by using [action groups](#). The feature can be enabled on both Azure Resource Manager and classic VMs for most scenarios.

NOTE

Before you install this version (v1), we would like you to know about the [next version](#), which is in preview right now. This new version (v2) offers all the same functionality as this one, but is designed to take advantage of newer technology in Azure. It adds some of the commonly requested features from customers, such as multi-subscription support from a single Start/Stop instance.

Start/Stop VMs during off-hours (v1) will deprecate on 5/21/2022.

This feature uses [Start-AzVm](#) cmdlet to start VMs. It uses [Stop-AzVM](#) for stopping VMs.

NOTE

Start/Stop VMs during off-hours has been updated to support the newest versions of the Azure modules that are available. The updated version of this feature, available in the Marketplace, doesn't support AzureRM modules because we have migrated from AzureRM to Az modules. While the runbooks have been updated to use the new Azure Az module cmdlets, they use the AzureRM prefix alias.

The feature provides a decentralized low-cost automation option for users who want to optimize their VM costs. You can use the feature to:

- [Schedule VMs to start and stop](#).
- Schedule VMs to start and stop in ascending order by [using Azure Tags](#). This activity is not supported for classic VMs.
- Autostop VMs based on [low CPU usage](#).

The following are limitations with the current feature:

- It manages VMs in any region, but can only be used in the same subscription as your Azure Automation account.
- It is available in Azure and Azure Government for any region that supports a Log Analytics workspace, an Azure Automation account, and alerts. Azure Government regions currently don't support email functionality.

Prerequisites

- The runbooks for the Start/Stop VMs during off hours feature work with an [Azure Run As account](#). The Run As account is the preferred authentication method because it uses certificate authentication instead of a password that might expire or change frequently.
- An [Azure Monitor Log Analytics workspace](#) that stores the runbook job logs and job stream results in a workspace to query and analyze. The Automation account and Log Analytics workspace need to be in the same subscription and supported region. The workspace needs to already exist, you cannot create a new workspace during deployment of this feature.

We recommend that you use a separate Automation account for working with VMs enabled for the Start/Stop VMs during off-hours feature. Azure module versions are frequently upgraded, and their parameters might change. The feature isn't upgraded on the same cadence and it might not work with newer versions of the cmdlets that it uses. Before importing the updated modules into your production Automation account(s), we recommend you import them into a test Automation account to verify there aren't any compatibility issues.

Permissions

You must have certain permissions to enable VMs for the Start/Stop VMs during off-hours feature. The permissions are different depending on whether the feature uses a pre-created Automation account and Log Analytics workspace or creates a new account and workspace.

You don't need to configure permissions if you're a Contributor on the subscription and a Global Administrator in your Azure Active Directory (AD) tenant. If you don't have these rights or need to configure a custom role, make sure that you have the permissions described below.

Permissions for pre-existing Automation account and Log Analytics workspace

To enable VMs for the Start/Stop VMs during off-hours feature using an existing Automation account and Log Analytics workspace, you need the following permissions on the Resource Group scope. To learn more about roles, see [Azure custom roles](#).

PERMISSION	SCOPE
Microsoft.Automation/automationAccounts/read	Resource Group
Microsoft.Automation/automationAccounts/variables/write	Resource Group
Microsoft.Automation/automationAccounts/schedules/write	Resource Group
Microsoft.Automation/automationAccounts/runbooks/write	Resource Group
Microsoft.Automation/automationAccounts/connections/write	Resource Group
Microsoft.Automation/automationAccounts/certificates/write	Resource Group
Microsoft.Automation/automationAccounts/modules/write	Resource Group
Microsoft.Automation/automationAccounts/modules/read	Resource Group
Microsoft.automation/automationAccounts/jobSchedules/write	Resource Group
Microsoft.Automation/automationAccounts/jobs/write	Resource Group
Microsoft.Automation/automationAccounts/jobs/read	Resource Group
Microsoft.OperationsManagement/solutions/write	Resource Group
Microsoft.OperationalInsights/workspaces/*	Resource Group
Microsoft.Insights/diagnosticSettings/write	Resource Group

PERMISSION	SCOPE
Microsoft.Insights/ActionGroups/Write	Resource Group
Microsoft.Insights/ActionGroups/read	Resource Group
Microsoft.Resources/subscriptions/resourceGroups/read	Resource Group
Microsoft.Resources/deployments/*	Resource Group

Permissions for new Automation account and new Log Analytics workspace

You can enable VMs for the Start/Stop VMs during off-hours feature using a new Automation account and Log Analytics workspace. In this case, you need the permissions defined in the previous section and the permissions defined in this section. You also require the following roles:

- Co-Administrator on subscription. This role is required to create the Classic Run As account if you are going to manage classic VMs. [Classic Run As accounts](#) are no longer created by default.
- Membership in the [Azure AD Application Developer](#) role. For more information on configuring Run As Accounts, see [Permissions to configure Run As accounts](#).
- Contributor on the subscription or the following permissions.

PERMISSION	SCOPE
Microsoft.Authorization/Operations/read	Subscription
Microsoft.Authorization/permissions/read	Subscription
Microsoft.Authorization/roleAssignments/read	Subscription
Microsoft.Authorization/roleAssignments/write	Subscription
Microsoft.Authorization/roleAssignments/delete	Subscription
Microsoft.Automation/automationAccounts/connections/read	Resource Group
Microsoft.Automation/automationAccounts/certificates/read	Resource Group
Microsoft.Automation/automationAccounts/write	Resource Group
Microsoft.OperationalInsights/workspaces/write	Resource Group

Components

The Start/Stop VMs during off-hours feature include preconfigured runbooks, schedules, and integration with Azure Monitor Logs. You can use these elements to tailor the startup and shutdown of your VMs to suit your business needs.

Runbooks

The following table lists the runbooks that the feature deploys to your Automation account. Do NOT make changes to the runbook code. Instead, write your own runbook for new functionality.

IMPORTANT

Don't directly run any runbook with **child** appended to its name.

All parent runbooks include the `WhatIf` parameter. When set to True, the parameter supports detailing the exact behavior the runbook takes when run without the parameter and validates that the correct VMs are targeted. A runbook only performs its defined actions when the `WhatIf` parameter is set to False.

RUNBOOK	PARAMETERS	DESCRIPTION
AutoStop_CreateAlert_Child	VMOBJECT AlertAction WebHookURI	Called from the parent runbook. This runbook creates alerts on a per-resource basis for the Auto-Stop scenario.
AutoStop_CreateAlert_Parent	VMLIST WhatIf: True or False	Creates or updates Azure alert rules on VMs in the targeted subscription or resource groups. <code>VMLIST</code> is a comma-separated list of VMs (with no whitespaces), for example, <code>vm1, vm2, vm3</code> . <code>WhatIf</code> enables validation of runbook logic without executing.
AutoStop_Disable	None	Disables Auto-Stop alerts and default schedule.
AutoStop_VM_Child	WebHookData	Called from the parent runbook. Alert rules call this runbook to stop a classic VM.
AutoStop_VM_Child_ARM	WebHookData	Called from the parent runbook. Alert rules call this runbook to stop a VM.
ScheduledStartStop_Base_Classic	CloudServiceName Action: Start or Stop VMLIST	Performs action start or stop in classic VM group by Cloud Services.
ScheduledStartStop_Child	VMName Action: Start or Stop ResourceGroupName	Called from the parent runbook. Executes a start or stop action for the scheduled stop.
ScheduledStartStop_Child_Classic	VMName Action: Start or Stop ResourceGroupName	Called from the parent runbook. Executes a start or stop action for the scheduled stop for classic VMs.
ScheduledStartStop_Parent	Action: Start or Stop VMLIST WhatIf: True or False	Starts or stops all VMs in the subscription. Edit the variables <code>External_Start_ResourceGroupNames</code> and <code>External_Stop_ResourceGroupNames</code> to only execute on these targeted resource groups. You can also exclude specific VMs by updating the <code>External_ExcludeVMNames</code> variable.

RUNBOOK	PARAMETERS	DESCRIPTION
SequencedStartStop_Parent	Action: Start or Stop WhatIf: True or False VMList	<p>Creates tags named <code>sequencestart</code> and <code>sequencestop</code> on each VM for which you want to sequence start/stop activity. These tag names are case-sensitive. The value of the tag should be a list of positive integers, for example, <code>1,2,3</code>, that corresponds to the order in which you want to start or stop.</p> <p>Note: VMs must be within resource groups defined in <code>External_Start_ResourceGroupNames</code>,</p> <p>,</p> <p><code>External_Stop_ResourceGroupNames</code>, and <code>External_ExcludeVMNames</code> variables. They must have the appropriate tags for actions to take effect.</p>

Variables

The following table lists the variables created in your Automation account. Only modify variables prefixed with `External`. Modifying variables prefixed with `Internal` causes undesirable effects.

NOTE

Limitations on VM name and resource group are largely a result of variable size. See [Variable assets in Azure Automation](#).

VARIABLE	DESCRIPTION
<code>External_AutoStop_Condition</code>	The conditional operator required for configuring the condition before triggering an alert. Acceptable values are <code>GreaterThan</code> , <code>GreaterThanOrEqual</code> , <code>LessThan</code> , and <code>LessThanOrEqual</code> .
<code>External_AutoStop_Description</code>	The alert to stop the VM if the CPU percentage exceeds the threshold.
<code>External_AutoStop_Frequency</code>	The evaluation frequency for rule. This parameter accepts input in timespan format. Possible values are from 5 minutes to 6 hours.
<code>External_AutoStop_MetricName</code>	The name of the performance metric for which the Azure Alert rule is to be configured.
<code>External_AutoStop_Severity</code>	Severity of the metric alert, which can range from 0 to 4.
<code>External_AutoStop_Threshold</code>	The threshold for the Azure Alert rule specified in the variable <code>External_AutoStop_MetricName</code> . Percentage values range from 1 to 100.
<code>External_AutoStop_TimeAggregationOperator</code>	The time aggregation operator applied to the selected window size to evaluate the condition. Acceptable values are <code>Average</code> , <code>Minimum</code> , <code>Maximum</code> , <code>Total</code> , and <code>Last</code> .

VARIABLE	DESCRIPTION
External_AutoStop_TimeWindow	The size of the window during which Azure analyzes selected metrics for triggering an alert. This parameter accepts input in timespan format. Possible values are from 5 minutes to 6 hours.
External_EnableClassicVMs	Value specifying if classic VMs are targeted by the feature. The default value is True. Set this variable to False for Azure Cloud Solution Provider (CSP) subscriptions. Classic VMs require a Classic Run As account .
External_ExcludeVMNames	Comma-separated list of VM names to exclude, limited to 140 VMs. If you add more than 140 VMs to the list, VMs specified for exclusion might be inadvertently started or stopped.
External_Start_ResourceGroupNames	Comma-separated list of one or more resource groups that are targeted for start actions.
External_Stop_ResourceGroupNames	Comma-separated list of one or more resource groups that are targeted for stop actions.
External_WaitTimeForVMRetrySeconds	The wait time in seconds for the actions to be performed on the VMs for the SequencedStartStop_Parent runbook. This variable allows the runbook to wait for child operations for a specified number of seconds before proceeding with the next action. The maximum wait time is 10800, or three hours. The default value is 2100 seconds.
Internal_AutomationAccountName	Specifies the name of the Automation account.
Internal_AutoSnooze_ARM_WebhookURI	The webhook URI called for the AutoStop scenario for VMs.
Internal_AutoSnooze_WebhookUri	The webhook URI called for the AutoStop scenario for classic VMs.
Internal_AzureSubscriptionId	The Azure subscription ID.
Internal_ResourceGroupName	The Automation account resource group name.

NOTE

For the variable `External_WaitTimeForVMRetryInSeconds`, the default value has been updated from 600 to 2100.

Across all scenarios, the variables `External_Start_ResourceGroupNames`, `External_Stop_ResourceGroupNames`, and `External_ExcludeVMNames` are necessary for targeting VMs, except for the comma-separated VM lists for the **AutoStop_CreateAlert_Parent**, **SequencedStartStop_Parent**, and **ScheduledStartStop_Parent** runbooks. That is, your VMs must belong to target resource groups for start and stop actions to occur. The logic works similar to Azure Policy, in that you can target the subscription or resource group and have actions inherited by newly created VMs. This approach avoids having to maintain a separate schedule for every VM and manage starts and stops in scale.

Schedules

The following table lists each of the default schedules created in your Automation account. You can modify them

or create your own custom schedules. By default, all schedules are disabled except for the **Scheduled_StartVM** and **Scheduled_StopVM** schedules.

Don't enable all schedules, because doing so might create overlapping schedule actions. It's best to determine which optimizations you want to do and modify them accordingly. See the example scenarios in the overview section for further explanation.

SCHEDULE NAME	FREQUENCY	DESCRIPTION
Schedule_AutoStop_CreateAlert_Parent	Every 8 hours	Runs the AutoStop_CreateAlert_Parent runbook every 8 hours, which in turn stops the VM-based values in <code>External_Start_ResourceGroupNames</code> , <code>External_Stop_ResourceGroupNames</code> , and <code>External_ExcludeVMNames</code> variables. Alternatively, you can specify a comma-separated list of VMs by using the <code>VMList</code> parameter.
Scheduled_StopVM	User-defined, daily	Runs the ScheduledStopStart_Parent runbook with a parameter of <code>stop</code> every day at the specified time. Automatically stops all VMs that meet the rules defined by variable assets. Enable the related schedule Scheduled-StartVM .
Scheduled_StartVM	User-defined, daily	Runs the ScheduledStopStart_Parent runbook with a parameter value of <code>Start</code> every day at the specified time. Automatically starts all VMs that meet the rules defined by variable assets. Enable the related schedule Scheduled-StopVM .
Sequenced-StopVM	1:00 AM (UTC), every Friday	Runs the Sequenced_StopStop_Parent runbook with a parameter value of <code>Stop</code> every Friday at the specified time. Sequentially (ascending) stops all VMs with a tag of SequenceStop defined by the appropriate variables. For more information on tag values and asset variables, see Runbooks . Enable the related schedule, Sequenced-StartVM .

SCHEDULE NAME	FREQUENCY	DESCRIPTION
Sequenced-StartVM	1:00 PM (UTC), every Monday	Runs the SequencedStopStart_Parent runbook with a parameter value of <code>Start</code> every Monday at the specified time. Sequentially (descending) starts all VMs with a tag of SequenceStart defined by the appropriate variables. For more information on tag values and variable assets, see Runbooks . Enable the related schedule, Sequenced-StopVM .

Use the feature with classic VMs

If you are using the Start/Stop VMs during off-hours feature for classic VMs, Automation processes all your VMs sequentially per cloud service. VMs are still processed in parallel across different cloud services.

For use of the feature with classic VMs, you need a Classic Run As account, which is not created by default. For instructions on creating a Classic Run As account, see [Create a Classic Run As account](#).

If you have more than 20 VMs per cloud service, here are some recommendations:

- Create multiple schedules with the parent runbook **ScheduledStartStop_Parent** and specifying 20 VMs per schedule.
- In the schedule properties, use the `VMLIST` parameter to specify VM names as a comma-separated list (no whitespaces).

Otherwise, if the Automation job for this feature runs more than three hours, it's temporarily unloaded or stopped per the [fair share](#) limit.

Azure CSP subscriptions support only the Azure Resource Manager model. Non-Azure Resource Manager services are not available in the program. When the Start/Stop VMs during off-hours feature runs, you might receive errors since it has cmdlets to manage classic resources. To learn more about CSP, see [Available services in CSP subscriptions](#). If you use a CSP subscription, you should set the `External_EnableClassicVMs` variable to `False` after deployment.

NOTE

This article was recently updated to use the term Azure Monitor logs instead of Log Analytics. Log data is still stored in a Log Analytics workspace and is still collected and analyzed by the same Log Analytics service. We are updating the terminology to better reflect the role of [logs in Azure Monitor](#). See [Azure Monitor terminology changes](#) for details.

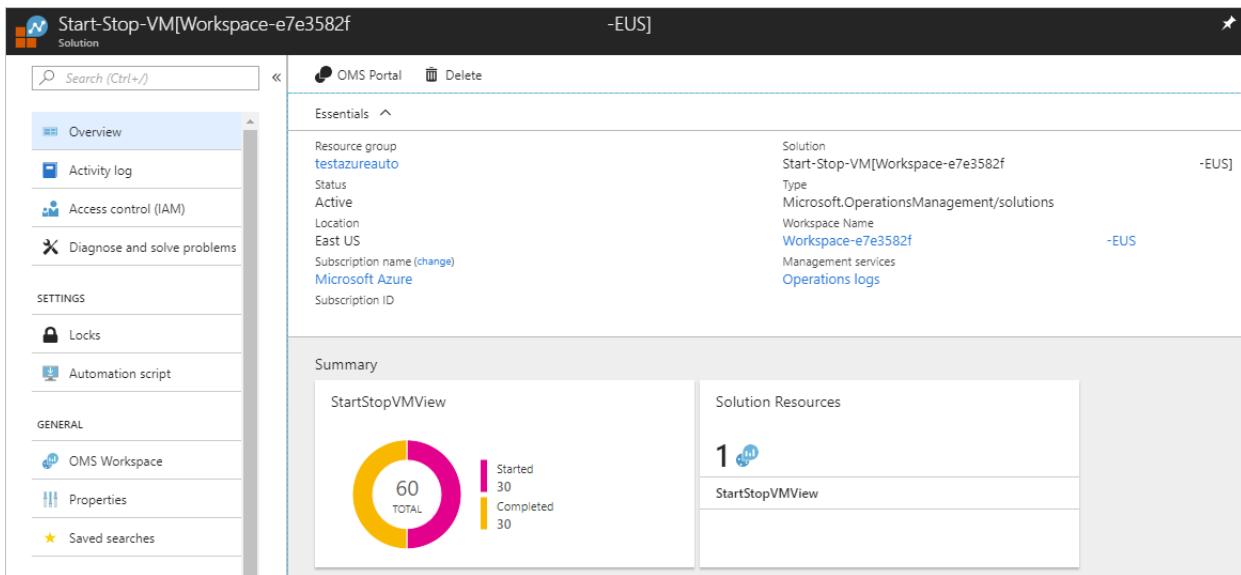
View the feature

Use one of the following mechanisms to access the enabled feature:

- From your Automation account, select **Start/Stop VM** under **Related Resources**. On the Start/Stop VM page, select **Manage the solution** under **Manage Start/Stop VM Solutions**.
- Navigate to the Log Analytics workspace linked to your Automation account. After selecting the workspace, choose **Solutions** from the left pane. On the Solutions page, select **Start-Stop-VM[workspace]** from the list.

Selecting the feature displays the **Start-Stop-VM[workspace]** page. Here you can review important details,

such as the information in the **StartStopVM** tile. As in your Log Analytics workspace, this tile displays a count and a graphical representation of the runbook jobs for the feature that have started and have finished successfully.



The screenshot shows the Azure OMS Portal interface for a solution named "Start-Stop-VM[Workspace-e7e3582f]". The left sidebar contains navigation links like Overview, Activity log, Access control (IAM), Diagnose and solve problems, Locks, Automation script, OMS Workspace, Properties, and Saved searches. The main content area has tabs for Essentials and Summary. The Essentials tab displays resource group "testazureauto", status "Active", location "East US", subscription "Microsoft Azure", and workspace "Workspace-e7e3582f". The Summary tab features a donut chart titled "StartStopVMView" showing 60 total jobs: 30 Started (purple) and 30 Completed (yellow). It also lists "Solution Resources" with one item: "StartStopVMView".

You can perform further analysis of the job records by clicking the donut tile. The dashboard shows job history and predefined log search queries. Switch to the log analytics advanced portal to search based on your search queries.

Update the feature

If you've deployed a previous version of Start/Stop VMs during off-hours, delete it from your account before deploying an updated release. Follow the steps to [remove the feature](#) and then follow the steps to [enable it](#).

Next steps

To enable the feature on VMs in your environment, see [Enable Start/Stop VMs during off-hours](#).

Supported regions for linked Log Analytics workspace

4/2/2021 • 3 minutes to read • [Edit Online](#)

In Azure Automation, you can enable the Update Management, Change Tracking and Inventory, and Start/Stop VMs during off-hours features for your servers and virtual machines. These features have a dependency on a Log Analytics workspace, and therefore require linking the workspace with an Automation account. However, only certain regions are supported to link them together. In general, the mapping is *not* applicable if you plan to link an Automation account to a workspace that won't have these features enabled.

The mappings discussed here apply only to linking the Log Analytics Workspace to an Automation account. They do not apply to the virtual machines (VMs) that are connected to the workspace that's linked to the Automation Account. VMs aren't limited to the regions supported by a given Log Analytics workspace. They can be in any region. Keep in mind that having the VMs in a different region may affect state, local, and country regulatory requirements, or your company's compliance requirements. Having VMs in a different region could also introduce data bandwidth charges.

Before connecting VMs to a workspace in a different region, you should review the requirements and potential costs to confirm and understand the legal and cost implications.

This article provides the supported mappings in order to successfully enable and use these features in your Automation account.

For more information, see [Log Analytics workspace and Automation account](#).

Supported mappings

NOTE

As shown in following table, only one mapping can exist between Log Analytics and Azure Automation.

The following table shows the supported mappings:

LOG ANALYTICS WORKSPACE REGION	AZURE AUTOMATION REGION
US	
EastUS ¹	EastUS2
EastUS ²	EastUS
WestUS	WestUS
WestUS2	WestUS2
NorthCentralUS	NorthCentralUS
CentralUS	CentralUS

LOG ANALYTICS WORKSPACE REGION	AZURE AUTOMATION REGION
SouthCentralUS	SouthCentralUS
WestCentralUS	WestCentralUS
Brazil	
BrazilSouth	BrazilSouth
Canada	
CanadaCentral	CanadaCentral
China	
ChinaEast2 ³	ChinaEast2
Asia Pacific	
EastAsia	EastAsia
SoutheastAsia	SoutheastAsia
India	
CentralIndia	CentralIndia
Japan	
JapanEast	JapanEast
Australia	
AustraliaEast	AustraliaEast
AustraliaSoutheast	AustraliaSoutheast
Korea	
KoreaCentral	KoreaCentral
Norway	
NorwayEast	NorwayEast
Europe	
NorthEurope	NorthEurope
WestEurope	WestEurope

LOG ANALYTICS WORKSPACE REGION	AZURE AUTOMATION REGION
France	
FranceCentral	FranceCentral
United Kingdom	
UKSouth	UKSouth
Switzerland	
SwitzerlandNorth	SwitzerlandNorth
United Arab Emirates	
UAENorth	UAENorth
US Gov	
USGovVirginia	USGovVirginia
USGovArizona ³	USGovArizona

¹ EastUS mapping for Log Analytics workspaces to Automation accounts isn't an exact region-to-region mapping, but is the correct mapping.

² EastUS2 mapping for Log Analytics workspaces to Automation accounts isn't an exact region-to-region mapping, but is the correct mapping.

³ In this region, only Update Management is supported, and other features like Change Tracking and Inventory are not available at this time.

Unlink a workspace

If you decide that you no longer want to integrate your Automation account with a Log Analytics workspace, you can unlink your account directly from the Azure portal. Before proceeding, you first need to [remove](#) Update Management, Change Tracking and Inventory, and Start/Stop VMs during off-hours if you are using them. If you don't remove them, you can't complete the unlinking operation.

With the features removed, you can follow the steps below to unlink your Automation account.

NOTE

Some features, including earlier versions of the Azure SQL monitoring solution, might have created Automation assets that need to be removed prior to unlinking the workspace.

- From the Azure portal, open your Automation account. On the Automation account page, select **Linked workspace** under **Related Resources**.
- On the Unlink workspace page, select **Unlink workspace**. You receive a prompt verifying if you want to continue.
- While Azure Automation is unlinking the account from your Log Analytics workspace, you can track the

progress under **Notifications** from the menu.

4. If you used Update Management, optionally you might want to remove the following items that are no longer needed:
 - Update schedules: Each has a name that matches an update deployment that you created.
 - Hybrid worker groups created for the feature: Each has a name similar to
`machine1.contoso.com_9ceb8108-26c9-4051-b6b3-227600d715c8`.
5. If you used Start/Stop VMs during off-hours, optionally you can remove the following items that are no longer needed:
 - Start and stop VM runbook schedules
 - Start and stop VM runbooks
 - Variables

Alternatively, you can unlink your workspace from your Automation account within the workspace.

1. In the workspace, select **Automation Account** under **Related Resources**.
2. On the Automation Account page, select **Unlink account**.

Next steps

- Learn about Update Management in [Update Management overview](#).
- Learn about Change Tracking and Inventory in [Change Tracking and Inventory overview](#).
- Learn about Start/Stop VMs during off-hours in [Start/Stop VMs during off-hours overview](#).

Enable Start/Stop VMs during off-hours

5/18/2021 • 4 minutes to read • [Edit Online](#)

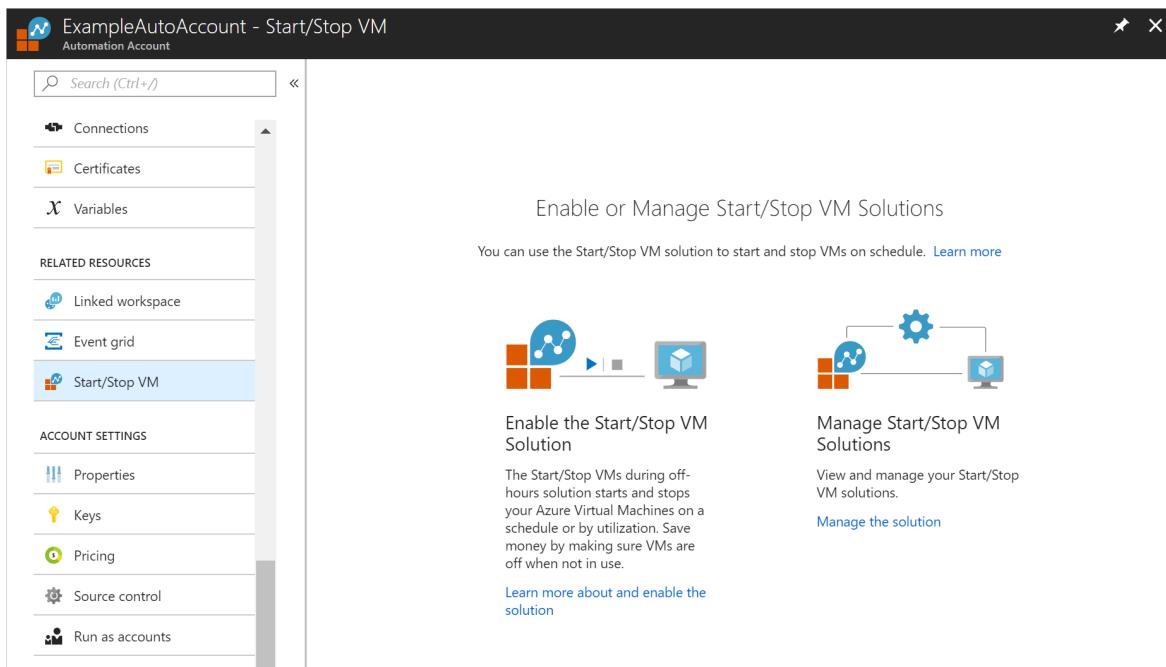
Perform the steps in this topic in sequence to enable the Start/Stop VMs during off-hours feature for VMs using a new or existing Automation account and linked Log Analytics workspace. After completing the setup process, configure the variables to customize the feature.

NOTE

To use this feature with classic VMs, you need a Classic Run As account, which is not created by default. See [Create a Classic Run As account](#).

Enable and configure

1. Sign in to the Azure [portal](#).
2. Search for and select **Automation Accounts**.
3. On the **Automation Accounts** page, select your Automation account from the list.
4. From the Automation account, select **Start/Stop VM** under **Related Resources**. From here, you can click **Learn more about and enable the solution**. If you already have the feature deployed, you can click **Manage the solution** and find it in the list.



NOTE

You can also create the resource from anywhere in the Azure portal, by clicking **Create a resource**. In the Marketplace page, type a keyword such as **Start** or **Start/Stop**. As you begin typing, the list filters based on your input. Alternatively, you can type in one or more keywords from the full name of the feature and then press **Enter**. Select **Start/Stop VMs during off-hours** from the search results.

5. On the Start/Stop VMs during off-hours page for the selected deployment, review the summary

information and then click **Create**.

Start/Stop VMs during off-hours

Microsoft

The Start/Stop VMs during off-hours solution starts and stops your Azure Virtual Machines on a schedule or by utilization. Save money by making sure VMs are off when not being used.

The Start/Stop VMs during off-hours solution relies on three Azure services:

- **Automation:** starts and stops your virtual machines on a schedule
- **Log Analytics:** visualizes the successful start and stop of your machines
- **Monitor:** alerts and email notification for VM state changes

You will be charged based on the pricing of the services above. Note that the logs for all runbooks jobs in the selected Automation account will be sent to Log Analytics as part of this solution.

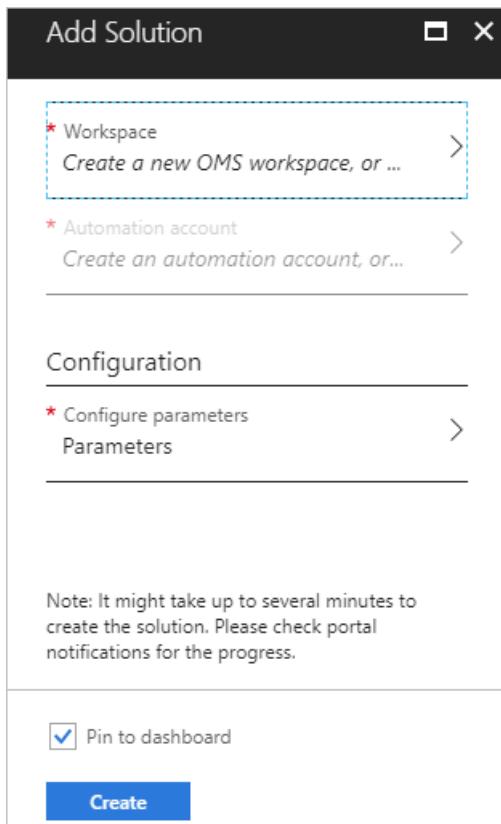
If you use this solution, it will only use automation job minutes and log ingestion. That is, this solution does not add additional OMS nodes to your environment.

[Save for later](#)

PUBLISHER	Microsoft
USEFUL LINKS	Azure Automation Log Analytics Send Azure log to Log Analytics Azure Monitor Azure Automation pricing Log Analytics pricing Azure Monitor pricing Documentation

[Create](#)

With the resource created, the Add Solution page appears. You're prompted to configure the feature before you can import it into your Automation account.



6. On the **Add Solution** page, select **Workspace**. Select an existing Log Analytics workspace from the list. If there isn't an Automation account in the same supported region as the workspace, you can create a new Automation account in the next step.

NOTE

When enabling features, only certain regions are supported for linking a Log Analytics workspace and an Automation account. For a list of the supported mapping pairs, see [Region mapping for Automation account and Log Analytics workspace](#).

7. On the **Add Solution** page if there isn't an Automation account available in the supported region as the workspace, select **Automation account**. You can create a new Automation account to associate with it by selecting **Create an Automation account**, and on the **Add Automation account** page, provide the the name of the Automation account in the **Name** field.

All other options are automatically populated, based on the Log Analytics workspace selected. You can't modify these options. An Azure Run As account is the default authentication method for the runbooks included with the feature.

After you click **OK**, the configuration options are validated and the Automation account is created. You can track its progress under **Notifications** from the menu.

8. On the Add Solution page, select **Configure parameters**. The **Parameters** page appears.

Parameters

Vm runbook

Target ResourceGroup Names (string) *

VM Exclude List (string)

Schedule

* Daily Start Time (datetime) *

* Daily Stop Time (datetime) *

Email functionality

Receive Email Notifications (string) *

Email Addresses (string)

OK

- Specify a value for the **Target ResourceGroup Names** field. The field defines group names that contain VMs for the feature to manage. You can enter more than one name and separate the names using commas (values are not case-sensitive). Using a wildcard is supported if you want to target VMs in all resource groups in the subscription. The values are stored in the `External_Start_ResourceGroupNames` and `External_Stop_ResourceGroupNames` variables.

IMPORTANT

The default value for **Target ResourceGroup Names** is a *. This setting targets all VMs in a subscription. If you don't want the feature to target all the VMs in your subscription, you must provide a list of resource group names before selecting a schedule.

- Specify a value for the **VM Exclude List (string)** field. This value is the name of one or more virtual machines from the target resource group. You can enter more than one name and separate the names using commas (values are not case-sensitive). Using a wildcard is supported. This value is stored in the `External_ExcludeVMNames` variable.
- Use the **Schedule** field to select a schedule for VM management by the feature. Select a start date and time for your schedule to create a recurring daily schedule starting at the chosen time. Selecting a different region is not available. To configure the schedule to your specific time zone after configuring the feature, see [Modify the startup and shutdown schedules](#).
- To receive email notifications from an [action group](#), accept the default value of **Yes** in the **Email notifications** field, and provide a valid email address. If you select **No** but decide at a later date that you want to receive email notifications, you can update the action group that is created with valid email addresses separated by commas. The following alert rules are created in the subscription:
 - `AutoStop_VM_Child`
 - `Scheduled_StartStop_Parent`

- Sequenced_StartStop_Parent

13. After you have configured the initial settings required for the feature, click **OK** to close the **Parameters** page.
14. Click **Create**. After all settings are validated, the feature deploys to your subscription. This process can take several seconds to finish, and you can track its progress under **Notifications** from the menu.

NOTE

If you have an Azure Cloud Solution Provider (Azure CSP) subscription, after deployment is complete, in your Automation account, go to **Variables** under **Shared Resources** and set the [External_EnableClassicVMs](#) variable to **False**. This stops the solution from looking for Classic VM resources.

Create alerts

Start/Stop VMs during off-hours doesn't include a predefined set of Automation job alerts. Review [Forward job data to Azure Monitor Logs](#) to learn about log data forwarded from the Automation account related to the runbook job results and how to create job failed alerts to support your DevOps or operational processes and procedures.

Next steps

- To set up the feature, see [Configure Stop/Start VMs during off-hours](#).
- To resolve feature errors, see [Troubleshoot Start/Stop VMs during off-hours issues](#).

Configure Start/Stop VMs during off-hours

4/28/2021 • 8 minutes to read • [Edit Online](#)

This article describes how to configure the [Start/Stop VMs during off-hours](#) feature to support the described scenarios. You can also learn how to:

- [Configure email notifications](#)
- [Add a VM](#)
- [Exclude a VM](#)
- [Modify the startup and shutdown schedules](#)

Scenario 1: Start/Stop VMs on a schedule

This scenario is the default configuration when you first deploy Start/Stop VMs during off-hours. For example, you can configure the feature to stop all VMs across a subscription when you leave work in the evening, and start them in the morning when you are back in the office. When you configure the schedules **Scheduled-StartVM** and **Scheduled-StopVM** during deployment, they start and stop targeted VMs.

Configuring the feature to just stop VMs is supported. See [Modify the startup and shutdown schedules](#) to learn how to configure a custom schedule.

NOTE

The time zone used by the feature is your current time zone when you configure the schedule time parameter. However, Azure Automation stores it in UTC format in Azure Automation. You don't have to do any time zone conversion, as this is handled during machine deployment.

To control the VMs that are in scope, configure the variables: `External_Start_ResourceGroupNames`, `External_Stop_ResourceGroupNames`, and `External_ExcludeVMNames`.

You can enable either targeting the action against a subscription and resource group, or targeting a specific list of VMs, but not both.

Target the start and stop actions against a subscription and resource group

1. Configure the `External_Stop_ResourceGroupNames` and `External_ExcludeVMNames` variables to specify the target VMs.
2. Enable and update the **Scheduled-StartVM** and **Scheduled-StopVM** schedules.
3. Run the **ScheduledStartStop_Parent** runbook with the **ACTION** parameter field set to **start** and the **WHATIF** parameter field set to True to preview your changes.

Target the start and stop action by VM list

1. Run the **ScheduledStartStop_Parent** runbook with **ACTION** set to **start**.
2. Add a comma-separated list of VMs (without spaces) in the **VMLIST** parameter field. An example list is `vm1,vm2,vm3`.
3. Set the **WHATIF** parameter field to True to preview your changes.
4. Configure the `External_ExcludeVMNames` variable with a comma-separated list of VMs (VM1,VM2,VM3), without spaces between comma-separated values.

5. This scenario does not honor the `External_Start_ResourceGroupNames` and `External_Stop_ResourceGroupNames` variables. For this scenario, you need to create your own Automation schedule. For details, see [Schedule a runbook in Azure Automation](#).

NOTE

The value for **Target ResourceGroup Names** is stored as the values for both `External_Start_ResourceGroupNames` and `External_Stop_ResourceGroupNames`. For further granularity, you can modify each of these variables to target different resource groups. For start action, use `External_Start_ResourceGroupNames`, and use `External_Stop_ResourceGroupNames` for stop action. VMs are automatically added to the start and stop schedules.

Scenario 2: Start/Stop VMs in sequence by using tags

In an environment that includes two or more components on multiple VMs supporting a distributed workload, supporting the sequence in which components are started and stopped in order is important.

Target the start and stop actions against a subscription and resource group

1. Add a `sequencestart` and a `sequencestop` tag with positive integer values to VMs that are targeted in `External_Start_ResourceGroupNames` and `External_Stop_ResourceGroupNames` variables. The start and stop actions are performed in ascending order. To learn how to tag a VM, see [Tag a Windows virtual machine in Azure](#) and [Tag a Linux virtual machine in Azure](#).
2. Modify the schedules **Sequenced-StartVM** and **Sequenced-StopVM** to the date and time that meet your requirements and enable the schedule.
3. Run the **SequencedStartStop_Parent** runbook with ACTION set to **start** and WHATIF set to True to preview your changes.
4. Preview the action and make any necessary changes before implementing against production VMs. When ready, manually execute the runbook with the parameter set to **False**, or let the Automation schedules **Sequenced-StartVM** and **Sequenced-StopVM** run automatically following your prescribed schedule.

Target the start and stop actions by VM list

1. Add a `sequencestart` and a `sequencestop` tag with positive integer values to VMs that you plan to add to the `VMList` parameter.
2. Run the **SequencedStartStop_Parent** runbook with ACTION set to **start**.
3. Add a comma-separated list of VMs (without spaces) in the `VMList` parameter field. An example list is `vm1,vm2,vm3`.
4. Set **WHATIF** to True to preview your changes.
5. Configure the `External_ExcludeVMNames` variable with a comma-separated list of VMs, without spaces between comma-separated values.
6. This scenario does not honor the `External_Start_ResourceGroupNames` and `External_Stop_ResourceGroupNames` variables. For this scenario, you need to create your own Automation schedule. For details, see [Schedule a runbook in Azure Automation](#).
7. Preview the action and make any necessary changes before implementing against production VMs. When ready, manually execute the **monitoring-and-diagnostics/monitoring-action-groupsrunbook** with the parameter set to **False**. Alternatively, let the Automation schedules **Sequenced-StartVM** and **Sequenced-StopVM** run automatically following your prescribed schedule.

Scenario 3: Start or stop automatically based on CPU utilization

Start/Stop VMs during off-hours can help manage the cost of running Azure Resource Manager and classic VMs in your subscription by evaluating machines that aren't used during non-peak periods, such as after hours, and automatically shutting them down if processor utilization is less than a specified percentage.

By default, the feature is pre-configured to evaluate the percentage CPU metric to see if average utilization is 5 percent or less. This scenario is controlled by the following variables and can be modified if the default values don't meet your requirements:

- `External_AutoStop_MetricName`
- `External_AutoStop_Threshold`
- `External_AutoStop_TimeAggregationOperator`
- `External_AutoStop_TimeWindow`
- `External_AutoStop_Frequency`
- `External_AutoStop_Severity`

You can enable and target the action against a subscription and resource group, or target a specific list of VMs.

When you run the **AutoStop_CreateAlert_Parent** runbook, it verifies that the targeted subscription, resource group(s), and VMs exist. If the VMs exist, the runbook calls the **AutoStop_CreateAlert_Child** runbook for each VM verified by the parent runbook. This child runbook:

- Creates a metric alert rule for each verified VM.
- Triggers the **AutoStop_VM_Child** runbook for a particular VM if the CPU drops below the configured threshold for the specified time interval.
- Attempts to stop the VM.

Target the autopstop action against all VMs in a subscription

1. Ensure that the `External_Stop_ResourceGroupNames` variable is empty or set to * (wildcard).
2. [Optional] If you want to exclude some VMs from the autopstop action, you can add a comma-separated list of VM names to the `External_ExcludeVMNames` variable.
3. Enable the **Schedule_AutoStop_CreateAlert_Parent** schedule to run to create the required Stop VM metric alert rules for all of the VMs in your subscription. Running this type of schedule lets you create new metric alert rules as new VMs are added to the subscription.

Target the autopstop action against all VMs in a resource group or multiple resource groups

1. Add a comma-separated list of resource group names to the `External_Stop_ResourceGroupNames` variable.
2. If you want to exclude some of the VMs from the autopstop, you can add a comma-separated list of VM names to the `External_ExcludeVMNames` variable.
3. Enable the **Schedule_AutoStop_CreateAlert_Parent** schedule to run to create the required Stop VM metric alert rules for all of the VMs in your resource groups. Running this operation on a schedule allows you to create new metric alert rules as new VMs are added to the resource group(s).

Target the autopstop action to a list of VMs

1. Create a new **schedule** and link it to the **AutoStop_CreateAlert_Parent** runbook, adding a comma-separated list of VM names to the `VMList` parameter.
2. Optionally, if you want to exclude some VMs from the autopstop action, you can add a comma-separated list of VM names (without spaces) to the `External_ExcludeVMNames` variable.

Configure email notifications

To change email notifications after Start/Stop VMs during off-hours is deployed, you can modify the action group created during deployment.

NOTE

Subscriptions in the Azure Government cloud don't support the email functionality of this feature.

1. In the Azure portal, click on **Alerts** under **Monitoring**, then **Manage actions**. On the **Manage actions** page, make sure you're on the **Action groups** tab. Select the action group called **StartStop_VM_Notification**.

Action group name	Short name	Resource group	Subscription	Status	Actions
Application Insights Sma...	SmartDetect	maic-rg	68627f8c-65b8-4601-b4...	Enabled	2 Email Azure Resource ...
StartStop_VM_Notification	StStAlert	prodwus	68627f8c-65b8-4601-b4...	Enabled	1 Email

2. On the **StartStop_VM_Notification** page, the **Basics** section will be filled in for you and can't be edited, except for the **Display name** field. Edit the name, or accept the suggested name. In the **Notifications** section, click the pencil icon to edit the action details. This opens the **Email/SMS message/Push/Voice** pane. Update the email address and click **OK** to save your changes.

Email/SMS message/Push/Voice

X

Add or edit an Email/SMS/Push/Voice action

Email

Email *

SMS (Carrier charges may apply)

Country code

1

▼

Phone number

Azure app Push Notifications

Azure account email ⓘ

Voice

Country code ⓘ

1

Phone number

Enable the common alert schema. [Learn more](#)

Yes

No

OK

You can add additional actions to the action group. To learn more about action groups, see [action groups](#)

The following is an example email that is sent when the feature shuts down virtual machines.

The screenshot shows a Microsoft Edge browser window with the title "Start/Stop VMs during off-hours Runbook: ScheduledStartStop_Parent has attempted an action - Message (HT...)" at the top. The main content is a message from "Azure Email Service No Reply Account" dated "Mon 6/11/2018 2:25 PM". The message subject is "Start/Stop VMs during off-hours Runbook: ScheduledStartStop_Parent has attempted an action". It is addressed to "Administrator". A note says "If there are problems with how this message is displayed, click here to view it in a web browser." Below the message, a blue bar says "We are notifying you because there are 1 counts of "ScheduledStartStop_Parent"". The message body contains the following details:

NAME	ScheduledStartStop_Parent
SEVERITY	Informational
RESOURCE	StartStopWorkspaceExample
SEARCH INTERVAL START TIME	6/11/2018 9:14:24 PM (UTC)
SEARCH INTERVAL DURATION	5 min
SEARCH QUERY	AzureDiagnostics where (RunbookName_s == "ScheduledStartStop_Parent") where (ResultDescription hasprefix "~") extend output = substring(ResultDescription,1) summarize by ResultDescription, output project output
SEARCH RESULTS	1 result(s)
DESCRIPTION	Start/Stop VMs during off-hours Runbook: ScheduledStartStop_Parent has attempted an action

Below the table, a light blue box contains the text "Top 1 result(s)" followed by the list of VMs: "Attempted the stop action on the following VMs: aks-agentpool-42602657-0 aks-agentpool-42602657-1 aks-agentpool-42602657-2 LinuxVM1 LinuxVM2 LinuxVM3 WinVM1 WinVM2 WinVM3 WinVM4 WinVM5 LinuxVM4 LinuxVM5 LinuxVM6 WinVM6".

Add or exclude VMs

The feature allows you to add VMs to be targeted or excluded.

Add a VM

There are two ways to ensure that a VM is included when the feature runs:

- Each of the parent [runbooks](#) of the feature has a `VMLIST` parameter. You can pass a comma-separated list of VM names (without spaces) to this parameter when scheduling the appropriate parent runbook for your situation, and these VMs will be included when the feature runs.
- To select multiple VMs, set `External_Start_ResourceGroupNames` and `External_Stop_ResourceGroupNames` with the resource group names that contain the VMs you want to start or stop. You can also set the variables to a value of `*` to have the feature run against all resource groups in the subscription.

Exclude a VM

To exclude a VM from Stop/start VMs during off-hours, you can add its name to the `External_ExcludeVMNames` variable. This variable is a comma-separated list of specific VMs (without spaces) to exclude from the feature. This list is limited to 140 VMs. If you add more than 140 VMs to this list, VMs that are set to be excluded might be inadvertently started or stopped.

Modify the startup and shutdown schedules

Managing the startup and shutdown schedules in this feature follows the same steps as outlined in [Schedule a runbook in Azure Automation](#). Separate schedules are required to start and stop VMs.

Configuring the feature to just stop VMs at a certain time is supported. In this scenario you just create a stop schedule and no corresponding start schedule.

1. Ensure that you've added the resource groups for the VMs to shut down in the

`External_Stop_ResourceGroupNames` variable.

2. Create your own schedule for the time when you want to shut down the VMs.
3. Navigate to the **ScheduledStartStop_Parent** runbook and click **Schedule**. This allows you to select the schedule you created in the preceding step.
4. Select **Parameters and run settings** and set the **ACTION** field to **Stop**.
5. Select **OK** to save your changes.

Next steps

- To monitor the feature during operation, see [Query logs from Start/Stop VMs during off-hours](#).
- To handle problems during VM management, see [Troubleshoot Start/Stop VMs during off-hours issues](#).

Query logs from Start/Stop VMs during off-hours

3/5/2021 • 3 minutes to read • [Edit Online](#)

Azure Automation forwards two types of records to the linked Log Analytics workspace: job logs and job streams. This article reviews the data available for [query](#) in Azure Monitor.

Job logs

PROPERTY	DESCRIPTION
Caller	Who initiated the operation. Possible values are either an email address or system for scheduled jobs.
Category	Classification of the type of data. For Automation, the value is JobLogs.
CorrelationId	GUID that is the Correlation ID of the runbook job.
JobId	GUID that is the ID of the runbook job.
operationName	Specifies the type of operation performed in Azure. For Automation, the value is Job.
resourceId	Specifies the resource type in Azure. For Automation, the value is the Automation account associated with the runbook.
ResourceGroup	Specifies the resource group name of the runbook job.
ResourceProvider	Specifies the Azure service that supplies the resources you can deploy and manage. For Automation, the value is Azure Automation.
ResourceType	Specifies the resource type in Azure. For Automation, the value is the Automation account associated with the runbook.
resultType	The status of the runbook job. Possible values are: <ul style="list-style-type: none">- Started- Stopped- Suspended- Failed- Succeeded
resultDescription	Describes the runbook job result state. Possible values are: <ul style="list-style-type: none">- Job is started- Job Failed- Job Completed
RunbookName	Specifies the name of the runbook.

PROPERTY	DESCRIPTION
SourceSystem	Specifies the source system for the data submitted. For Automation, the value is OpsManager
StreamType	Specifies the type of event. Possible values are: - Verbose - Output - Error - Warning
SubscriptionId	Specifies the subscription ID of the job.
Time	Date and time when the runbook job executed.

Job streams

PROPERTY	DESCRIPTION
Caller	Who initiated the operation. Possible values are either an email address or system for scheduled jobs.
Category	Classification of the type of data. For Automation, the value is JobStreams.
JobId	GUID that is the ID of the runbook job.
operationName	Specifies the type of operation performed in Azure. For Automation, the value is Job.
ResourceGroup	Specifies the resource group name of the runbook job.
resourceId	Specifies the resource ID in Azure. For Automation, the value is the Automation account associated with the runbook.
ResourceProvider	Specifies the Azure service that supplies the resources you can deploy and manage. For Automation, the value is Azure Automation.
ResourceType	Specifies the resource type in Azure. For Automation, the value is the Automation account associated with the runbook.
resultType	The result of the runbook job at the time the event was generated. A possible value is: - InProgress
resultDescription	Includes the output stream from the runbook.
RunbookName	The name of the runbook.
SourceSystem	Specifies the source system for the data submitted. For Automation, the value is OpsManager.

PROPERTY	DESCRIPTION
StreamType	The type of job stream. Possible values are: - Progress - Output - Warning - Error - Debug - Verbose
Time	Date and time when the runbook job executed.

When you perform any log search that returns category records of **JobLogs** or **JobStreams**, you can select the **JobLogs** or **JobStreams** view, which displays a set of tiles summarizing the updates returned by the search.

Sample log searches

The following table provides sample log searches for job records collected by Start/Stop VMs during off-hours.

QUERY	DESCRIPTION
Find jobs for runbook ScheduledStartStop_Parent that have finished successfully	<pre>search Category == "JobLogs" where (RunbookName_s == "ScheduledStartStop_Parent") where (ResultType == "Completed") summarize AggregatedValue = count() by ResultType, bin(TimeGenerated, 1h) sort by TimeGenerated desc</pre>
Find jobs for runbook ScheduledStartStop_Parent that have not completed successfully	<pre>search Category == "JobLogs" where (RunbookName_s == "ScheduledStartStop_Parent") where (ResultType == "Failed") summarize AggregatedValue = count() by ResultType, bin(TimeGenerated, 1h) sort by TimeGenerated desc</pre>
Find jobs for runbook SequencedStartStop_Parent that have finished successfully	<pre>search Category == "JobLogs" where (RunbookName_s == "SequencedStartStop_Parent") where (ResultType == "Completed") summarize AggregatedValue = count() by ResultType, bin(TimeGenerated, 1h) sort by TimeGenerated desc</pre>
Find jobs for runbook SequencedStartStop_Parent that have not completed successfully	<pre>search Category == "JobLogs" where (RunbookName_s == "SequencedStartStop_Parent") where (ResultType == "Failed") summarize AggregatedValue = count() by ResultType, bin(TimeGenerated, 1h) sort by TimeGenerated desc</pre>

Next steps

- To set up the feature, see [Configure Stop/Start VMs during off-hours](#).
- For information on log alerts during feature deployment, see [Create log alerts with Azure Monitor](#).
- To resolve feature errors, see [Troubleshoot Start/Stop VMs during off-hours issues](#).

Remove Start/Stop VMs during off-hours from Automation account

4/16/2021 • 3 minutes to read • [Edit Online](#)

After you enable the Start/Stop VMs during off-hours feature to manage the running state of your Azure VMs, you may decide to stop using it. Removing this feature can be done using one of the following methods based on the supported deployment models:

- Delete the resource group containing the Automation account and linked Azure Monitor Log Analytics workspace, each dedicated to support this feature.
- Unlink the Log Analytics workspace from the Automation account and delete the Automation account dedicated for this feature.
- Delete the feature from an Automation account and linked workspace that are supporting other management and monitoring objectives.

Deleting this feature only removes the associated runbooks, it doesn't delete the schedules or variables that were created during deployment or any custom-defined ones created after.

NOTE

Before proceeding, verify there aren't any [Resource Manager locks](#) applied at the subscription, resource group, or resource which prevents accidental deletion or modification of critical resources. When you deploy the Start/Stop VMs during off-hours solution, it sets the lock level to **CanNotDelete** against several dependent resources in the Automation account (specifically its runbooks and variables). Any locks need to be removed before you can delete the Automation account.

Delete the dedicated resource group

To delete the resource group, follow the steps outlined in the [Azure Resource Manager resource group and resource deletion](#) article.

Delete the Automation account

To delete your Automation account dedicated to Start/Stop VMs during off-hours, perform the following steps.

1. Sign in to Azure at <https://portal.azure.com>.
2. Navigate to your Automation account, and select **Linked workspace** under **Related resources**.
3. Select **Go to workspace**.
4. Click **Solutions** under **General**.
5. On the Solutions page, select **Start-Stop-VM[Workspace]**.
6. On the **VMMManagementSolution[Workspace]** page, select **Delete** from the menu.
7. While the information is verified and the feature is deleted, you can track the progress under **Notifications**, chosen from the menu. You're returned to the Solutions page after the removal process.

Unlink workspace from Automation account

There are two options for unlinking the Log Analytics workspace from your Automation account. You can perform this process from the Automation account or from the linked workspace.

To unlink from your Automation account, perform the following steps.

1. In the Azure portal, select **Automation Accounts**.
2. Open your Automation account and select **Linked workspace** under **Related Resources** on the left.
3. On the **Unlink workspace** page, select **Unlink workspace** and respond to prompts.

The screenshot shows the Azure portal interface for managing an Automation Account. The main title is "MAIC-AA-Pri | Linked workspace". On the left, there's a sidebar with "Related Resources" where "Linked workspace" is selected. The main content area shows a message: "This Automation account is linked to the following Log Analytics workspace: maic-la". It includes a "Go to workspace" button and a "Unlink workspace" button. Below this, it says: "To unlink this Automation account and the Log Analytics workspace you must first remove These are the following:" followed by a bulleted list: "• Update Management", "• Change Tracking", and "• Start/Stop VMs during off-hours". It continues: "After you remove these solutions you can click **Unlink workspace** above to complete the process". There's also a note about "Update Management" and "View schedules".

While it attempts to unlink the Log Analytics workspace, you can track the progress under **Notifications** from the menu.

To unlink from the workspace, perform the following steps.

1. In the Azure portal, select **Log Analytics workspaces**.
2. From the workspace, select **Automation Account** under **Related Resources**.
3. On the Automation Account page, select **Unlink account** and respond to prompts.

While it attempts to unlink the Automation account, you can track the progress under **Notifications** from the menu.

Delete Automation account

1. In the Azure portal, select **Automation Accounts**.
2. Open your Automation account and select **Delete** from the menu.

While the information is verified and the account is deleted, you can track the progress under **Notifications**, chosen from the menu.

Delete the feature

To delete Start/Stop VMs during off-hours from your Automation account, perform the following steps. The Automation account and Log Analytics workspace aren't deleted as part of this process. If you don't want to keep the Log Analytics workspace, you must manually delete it. For more information about deleting your workspace, see [Delete and recover Azure Log Analytics workspace](#).

1. Navigate to your Automation account, and select **Linked workspace** under **Related resources**.

2. Select **Go to workspace**.
3. Click **Solutions** under **General**.
4. On the Solutions page, select **Start-Stop-VM[Workspace]**.
5. On the **VMMManagementSolution[Workspace]** page, select **Delete** from the menu.

The screenshot shows the Azure portal interface for managing a solution named 'Start-Stop-VM[Workspace-e7e3582f]'. The left sidebar has a navigation bar with 'Search (Ctrl+ /)', 'Overview' (selected), 'Activity log', 'Access control (IAM)', 'Diagnose and solve problems', 'SETTINGS', and 'Locks'. The main content area is titled '-EUS' and contains the following details:

	Value
Solution	Start-Stop-VM[Workspace-e7e3582f]
Type	Microsoft.OperationsManagement/solutions
Resource group	testazureauto
Status	Active
Location	East US
Subscription name (change)	Microsoft Azure
Workspace Name	Workspace-e7e3582f
Management services	Operations logs
Subscription ID	-EUS

6. In the Delete Solution window, confirm that you want to delete the feature.
7. While the information is verified and the feature is deleted, you can track the progress under **Notifications**, chosen from the menu. You're returned to the Solutions page after the removal process.
8. If you don't want to keep the **resources** created by the feature or by you afterwards (such as, variables, schedules, etc.), you have to manually delete them from the account.

Next steps

To re-enable this feature, see [Enable Start/Stop during off-hours](#).

Troubleshoot Start/Stop VMs during off-hours issues

4/22/2021 • 8 minutes to read • [Edit Online](#)

This article provides information on troubleshooting and resolving issues that arise when you deploy the Azure Automation Start/Stop VMs during off-hours feature on your VMs.

Scenario: Start/Stop VMs during off-hours fails to properly deploy

Issue

When you deploy [Start/Stop VMs during off-hours](#), you receive one of the following errors:

```
Account already exists in another resourcegroup in a subscription. ResourceGroupName: [MyResourceGroup].
```

```
Resource 'StartStop_VM_Notification' was disallowed by policy. Policy identifiers: '[{}\\\"policyAssignment\\\":{}\\\"name\\\":\\\"[MyPolicyName]\\\"].'
```

```
The subscription is not registered to use namespace 'Microsoft.OperationsManagement'.
```

```
The subscription is not registered to use namespace 'Microsoft.Insights'.
```

```
The scope '/subscriptions/000000000000-0000-0000-0000-00000000/resourcegroups/<ResourceGroupName>/providers/Microsoft.OperationalInsights/workspaces/<WorkspaceName>/views/StartStopVMView' cannot perform write operation because following scope(s) are locked: '/subscriptions/000000000000-0000-0000-0000-00000000/resourceGroups/<ResourceGroupName>/providers/Microsoft.OperationalInsights/workspaces/<WorkspaceName>/views/StartStopVMView'. Please remove the lock and try again
```

```
A parameter cannot be found that matches parameter name 'TagName'
```

```
Start-AzureRmVm : Run Login-AzureRmAccount to login
```

Cause

Deployments can fail because of one of the following reasons:

- There's already an Automation account with the same name in the region selected.
- A policy disallows the deployment of Start/Stop VMs during off-hours.
- The `Microsoft.OperationsManagement`, `Microsoft.Insights`, or `Microsoft.Automation` resource type isn't registered.
- Your Log Analytics workspace is locked.
- You have an outdated version of the AzureRM modules or the Start/Stop VMs during off-hours feature.

Resolution

Review the following fixes for potential resolutions:

- Automation accounts need to be unique within an Azure region, even if they're in different resource

groups. Check your existing Automation accounts in the target region.

- An existing policy prevents a resource that's required for Start/Stop VMs during off-hours to be deployed. Go to your policy assignments in the Azure portal, and check whether you have a policy assignment that disallows the deployment of this resource. To learn more, see [RequestDisallowedByPolicy error](#).
- To deploy Start/Stop VMs during off-hours, your subscription needs to be registered to the following Azure resource namespaces:
 - Microsoft.OperationsManagement
 - Microsoft.Insights
 - Microsoft.Automation

To learn more about errors when you register providers, see [Resolve errors for resource provider registration](#).

- If you have a lock on your Log Analytics workspace, go to your workspace in the Azure portal and remove any locks on the resource.
- If these resolutions don't solve your issue, follow the instructions under [Update the feature](#) to redeploy Start/Stop VMs during off-hours.

Scenario: All VMs fail to start or stop

Issue

You've configured Start/Stop VMs during off-hours, but it doesn't start or stop all the VMs.

Cause

This error can be caused by one of the following reasons:

- A schedule isn't configured correctly.
- The Run As account might not be configured correctly.
- A runbook might have run into errors.
- The VMs might have been excluded.

Resolution

Review the following list for potential resolutions:

- Check that you've properly configured a schedule for Start/Stop VMs during off-hours. To learn how to configure a schedule, see [Schedules](#).
- Check the [job streams](#) to look for any errors. Look for jobs from one of the following runbooks:
 - AutoStop_CreateAlert_Child
 - AutoStop_CreateAlert_Parent
 - AutoStop_Disable
 - AutoStop_VM_Child
 - ScheduledStartStop_Base_Classic
 - ScheduledStartStop_Child_Classic
 - ScheduledStartStop_Child
 - ScheduledStartStop_Parent
 - SequencedStartStop_Parent
- Verify that your [Run As account](#) has proper permissions to the VMs you're trying to start or stop. To learn how to check the permissions on a resource, see [Quickstart: View roles assigned to a user using the Azure portal](#). You'll need to provide the application ID for the service principal used by the Run As

account. You can retrieve this value by going to your Automation account in the Azure portal. Select **Run as accounts** under **Account Settings**, and select the appropriate Run As account.

- VMs might not be started or stopped if they're being explicitly excluded. Excluded VMs are set in the `External_ExcludeVMNames` variable in the Automation account to which the feature is deployed. The following example shows how you can query that value with PowerShell.

```
Get-AzAutomationVariable -Name External_ExcludeVMNames -AutomationAccountName <automationAccountName>
-ResourceGroupName <resourceGroupName> | Select-Object Value
```

Scenario: Some of my VMs fail to start or stop

Issue

You've configured Start/Stop VMs during off-hours, but it doesn't start or stop some of the VMs configured.

Cause

This error can be caused by one of the following reasons:

- In the sequence scenario, a tag might be missing or incorrect.
- The VM might be excluded.
- The Run As account might not have enough permissions on the VM.
- The VM can have an issue that stopped it from starting or stopping.

Resolution

Review the following list for potential resolutions:

- When you use the [sequence scenario](#) of Start/Stop VMs during off-hours, you must make sure that each VM you want to start or stop has the proper tag. Make sure the VMs that you want to start have the `sequencestart` tag and the VMs you want to stop have the `sequencestop` tag. Both tags require a positive integer value. You can use a query similar to the following example to look for all the VMs with the tags and their values.

```
Get-AzResource | ? {$_ .Tags.Keys -contains "SequenceStart" -or $_ .Tags.Keys -contains "SequenceStop"}
| ft Name,Tags
```

- VMs might not be started or stopped if they're being explicitly excluded. Excluded VMs are set in the `External_ExcludeVMNames` variable in the Automation account to which the feature is deployed. The following example shows how you can query that value with PowerShell.

```
Get-AzAutomationVariable -Name External_ExcludeVMNames -AutomationAccountName <automationAccountName>
-ResourceGroupName <resourceGroupName> | Select-Object Value
```

- To start and stop VMs, the Run As account for the Automation account must have appropriate permissions to the VM. To learn how to check the permissions on a resource, see [Quickstart: View roles assigned to a user using the Azure portal](#). You'll need to provide the application ID for the service principal used by the Run As account. You can retrieve this value by going to your Automation account in the Azure portal. Select **Run as accounts** under **Account Settings** and select the appropriate Run As account.
- If the VM is having a problem starting or deallocated, there might be an issue on the VM itself. Examples are an update that's being applied when the VM is trying to shut down, a service that hangs, and more. Go to your VM resource, and check **Activity Logs** to see if there are any errors in the logs. You might also attempt to log in to the VM to see if there are any errors in the event logs. To learn more about

troubleshooting your VM, see [Troubleshooting Azure virtual machines](#).

- Check the [job streams](#) to look for any errors. In the portal, go to your Automation account and select **Jobs** under **Process Automation**.

Scenario: My custom runbook fails to start or stop my VMs

Issue

You've authored a custom runbook or downloaded one from the PowerShell Gallery, and it isn't working properly.

Cause

There can be many causes for the failure. Go to your Automation account in the Azure portal, and select **Jobs** under **Process Automation**. From the **Jobs** page, look for jobs from your runbook to view any job failures.

Resolution

We recommend that you:

- Use [Start/Stop VMs during off-hours](#) to start and stop VMs in Azure Automation.
- Be aware that Microsoft doesn't support custom runbooks. You might find a resolution for your custom runbook in [Troubleshoot runbook issues](#). Check the [job streams](#) to look for any errors.

Scenario: VMs don't start or stop in the correct sequence

Issue

The VMs that you've enabled for the feature don't start or stop in the correct sequence.

Cause

This issue is caused by incorrect tagging on the VMs.

Resolution

Follow these steps to ensure that the feature is enabled correctly:

1. Ensure that all VMs to be started or stopped have a `sequencestart` or `sequencestop` tag, depending on your situation. These tags need a positive integer as the value. VMs are processed in ascending order based on this value.
2. Make sure that the resource groups for the VMs to be started or stopped are in the `External_Start_ResourceGroupNames` or `External_Stop_ResourceGroupNames` variables, depending on your situation.
3. Test your changes by executing the `SequencedStartStop_Parent` runbook with the `WHATIF` parameter set to True to preview your changes.

Scenario: Start/Stop VMs during off-hours job fails with 403 forbidden error

Issue

You find jobs that failed with a `403 forbidden` error for Start/Stop VMs during off-hours runbooks.

Cause

This issue can be caused by an improperly configured or expired Run As account. It might also be because of inadequate permissions to the VM resources by the Run As account.

Resolution

To verify that your Run As account is properly configured, go to your Automation account in the Azure portal

and select **Run as accounts** under **Account Settings**. If a Run As account is improperly configured or expired, the status shows the condition.

If your Run As account is misconfigured, delete and re-create your Run As account. For more information, see [Azure Automation Run As accounts](#).

If the certificate is expired for your Run As account, follow the steps in [Self-signed certificate renewal](#) to renew the certificate.

If there are missing permissions, see [Quickstart: View roles assigned to a user using the Azure portal](#). You must provide the application ID for the service principal used by the Run As account. You can retrieve this value by going to your Automation account in the Azure portal. Select **Run as accounts** under **Account Settings**, and select the appropriate Run As account.

Scenario: My problem isn't listed here

Issue

You experience an issue or unexpected result when you use Start/Stop VMs during off-hours that isn't listed on this page.

Cause

Many times errors can be caused by using an old and outdated version of the feature.

NOTE

The Start/Stop VMs during off-hours feature has been tested with the Azure modules that are imported into your Automation account when you deploy the feature on VMs. The feature currently doesn't work with newer versions of the Azure module. This restriction only affects the Automation account that you use to run Start/Stop VMs during off-hours. You can still use newer versions of the Azure module in your other Automation accounts, as described in [Update Azure PowerShell modules](#).

Resolution

To resolve many errors, remove and update Start/Stop VMs during off-hours. You also can check the [job streams](#) to look for any errors.

Next steps

If you don't see your problem here or you can't resolve your issue, try one of the following channels for additional support:

- Get answers from Azure experts through [Azure Forums](#).
- Connect with [@AzureSupport](#), the official Microsoft Azure account for improving customer experience. Azure Support connects the Azure community to answers, support, and experts.
- File an Azure support incident. Go to the [Azure support site](#), and select **Get Support**.

Update Management overview

6/24/2021 • 12 minutes to read • [Edit Online](#)

You can use Update Management in Azure Automation to manage operating system updates for your Windows and Linux virtual machines in Azure, physical or VMs in on-premises environments, and in other cloud environments. You can quickly assess the status of available updates and manage the process of installing required updates for your machines reporting to Update Management.

As a service provider, you may have onboarded multiple customer tenants to [Azure Lighthouse](#). Update Management can be used to assess and schedule update deployments to machines in multiple subscriptions in the same Azure Active Directory (Azure AD) tenant, or across tenants using Azure Lighthouse.

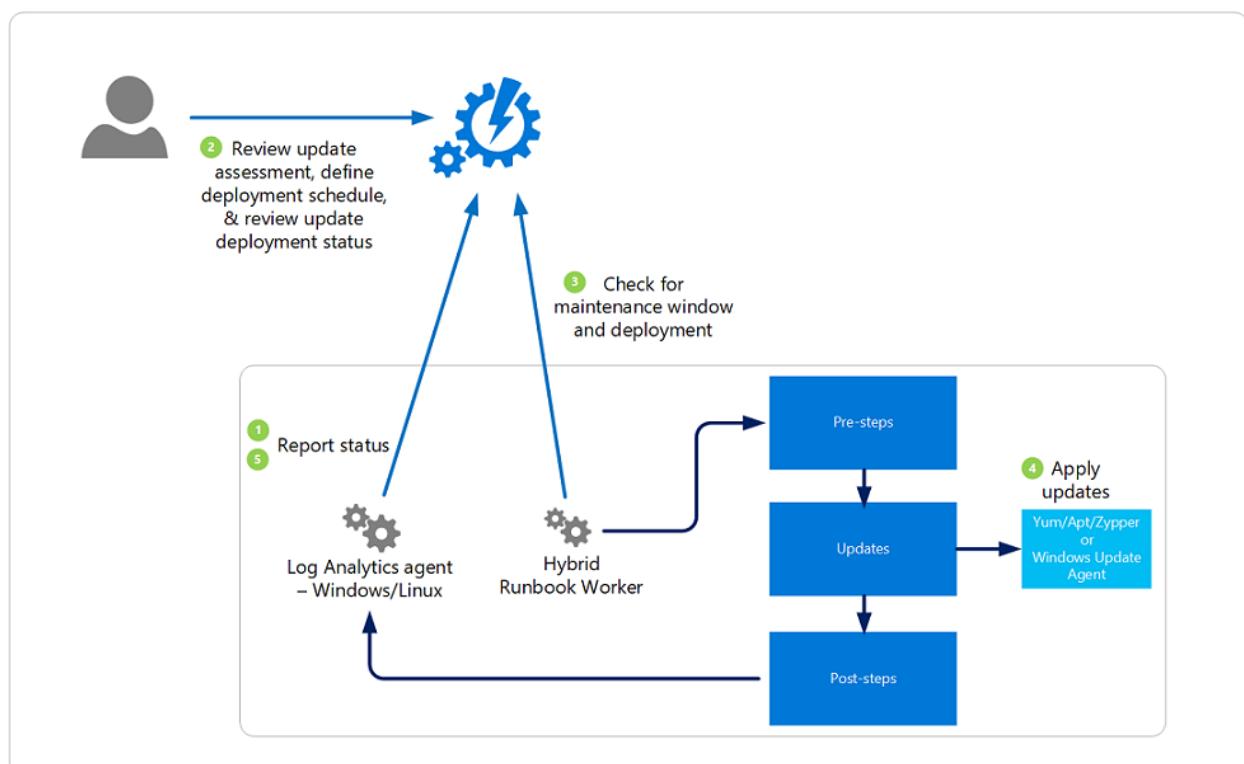
Microsoft offers other capabilities to help you manage updates for your Azure VMs or Azure virtual machine scale sets that you should consider as part of your overall update management strategy.

- If you are interested in automatically assessing and updating your Azure virtual machines to maintain security compliance with *Critical* and *Security* updates released each month, review [Automatic VM guest patching](#) (preview). This is an alternative update management solution for your Azure VMs to auto-update them during off-peak hours, including VMs within an availability set, compared to managing update deployments to those VMs from Update Management in Azure Automation.
- If you manage Azure virtual machine scale sets, review how to perform [automatic OS image upgrades](#) to safely and automatically upgrade the OS disk for all instances in the scale set.

Before deploying Update Management and enabling your machines for management, make sure that you understand the information in the following sections.

About Update Management

The following diagram illustrates how Update Management assesses and applies security updates to all connected Windows Server and Linux servers.



Update Management integrates with Azure Monitor Logs to store update assessments and update deployment results as log data, from assigned Azure and non-Azure machines. To collect this data, the Automation Account and Log Analytics workspace are linked together, and the Log Analytics agent for Windows and Linux is required on the machine and configured to report to this workspace. Update Management supports collecting information about system updates from agents in a System Center Operations Manager management group connected to the workspace. Having a machine registered for Update Management in more than one Log Analytics workspace (also referred to as multihoming) isn't supported.

The following table summarizes the supported connected sources with Update Management.

CONNECTED SOURCE	SUPPORTED	DESCRIPTION
Windows	Yes	Update Management collects information about system updates from Windows machines with the Log Analytics agent and installation of required updates.
Linux	Yes	Update Management collects information about system updates from Linux machines with the Log Analytics agent and installation of required updates on supported distributions.
Operations Manager management group	Yes	Update Management collects information about software updates from agents in a connected management group. A direct connection from the Operations Manager agent to Azure Monitor logs isn't required. Log data is forwarded from the management group to the Log Analytics workspace.

The machines assigned to Update Management report how up to date they are based on what source they are configured to synchronize with. Windows machines can be configured to report to Windows Server Update Services or Microsoft Update, and Linux machines can be configured to report to a local or public repo. You can also use Update Management with Microsoft Endpoint Configuration Manager, and to learn more see [Integrate Update Management with Windows Endpoint Configuration Manager](#).

If the Windows Update Agent (WUA) on the Windows machine is configured to report to WSUS, depending on when WSUS last synchronized with Microsoft Update, the results might differ from what Microsoft Update shows. This behavior is the same for Linux machines that are configured to report to a local repo instead of a public repo. On a Windows machine, the compliance scan is run every 12 hours by default. For a Linux machine, the compliance scan is performed every hour by default. If the Log Analytics agent is restarted, a compliance scan is started within 15 minutes. When a machine completes a scan for update compliance, the agent forwards the information in bulk to Azure Monitor Logs.

You can deploy and install software updates on machines that require the updates by creating a scheduled deployment. Updates classified as *Optional* aren't included in the deployment scope for Windows machines. Only required updates are included in the deployment scope.

The scheduled deployment defines which target machines receive the applicable updates. It does so either by explicitly specifying certain machines or by selecting a [computer group](#) that's based on log searches of a specific set of machines (or based on an [Azure query](#) that dynamically selects Azure VMs based on specified criteria). These groups differ from [scope configuration](#), which is used to control the targeting of machines that receive the

configuration to enable Update Management. This prevents them from performing and reporting update compliance, and install approved required updates.

While defining a deployment, you also specify a schedule to approve and set a time period during which updates can be installed. This period is called the maintenance window. A 20-minute span of the maintenance window is reserved for reboots, assuming one is needed and you selected the appropriate reboot option. If patching takes longer than expected and there's less than 20 minutes in the maintenance window, a reboot won't occur.

After an update package is scheduled for deployment, it takes 2 to 3 hours for the update to show up for Linux machines for assessment. For Windows machines, it takes 12 to 15 hours for the update to show up for assessment after it's been released. Before and after update installation, a scan for update compliance is performed and the log data results is forwarded to the workspace.

Updates are installed by runbooks in Azure Automation. You can't view these runbooks, and they don't require any configuration. When an update deployment is created, it creates a schedule that starts a master update runbook at the specified time for the included machines. The master runbook starts a child runbook on each agent that initiates the installation of the required updates with the Windows Update agent on Windows, or the applicable command on supported Linux distro.

At the date and time specified in the update deployment, the target machines execute the deployment in parallel. Before installation, a scan is run to verify that the updates are still required. For WSUS client machines, if the updates aren't approved in WSUS, update deployment fails.

Limits

For limits that apply to Update Management, see [Azure Automation service limits](#).

Permissions

To create and manage update deployments, you need specific permissions. To learn about these permissions, see [Role-based access - Update Management](#).

Update Management components

Update Management uses the resources described in this section. These resources are automatically added to your Automation account when you enable Update Management.

Hybrid Runbook Worker groups

After you enable Update Management, any Windows machine that's directly connected to your Log Analytics workspace is automatically configured as a system Hybrid Runbook Worker to support the runbooks that support Update Management.

Each Windows machine that's managed by Update Management is listed in the Hybrid worker groups pane as a System hybrid worker group for the Automation account. The groups use the `Hostname_FQDN_GUID` naming convention. You can't target these groups with runbooks in your account. If you try, the attempt fails. These groups are intended to support only Update Management. To learn more about viewing the list of Windows machines configured as a Hybrid Runbook Worker, see [view Hybrid Runbook Workers](#).

You can add the Windows machine to a user Hybrid Runbook Worker group in your Automation account to support Automation runbooks if you use the same account for Update Management and the Hybrid Runbook Worker group membership. This functionality was added in version 7.2.12024.0 of the Hybrid Runbook Worker.

Management packs

The following management packs are installed on the machines managed by Update Management. If your Operations Manager management group is [connected to a Log Analytics workspace](#), the management packs are

installed in the Operations Manager management group. You don't need to configure or manage these management packs.

- Microsoft System Center Advisor Update Assessment Intelligence Pack
(Microsoft.IntelligencePacks.UpdateAssessment)
- Microsoft.IntelligencePack.UpdateAssessment.Configuration
(Microsoft.IntelligencePack.UpdateAssessment.Configuration)
- Update Deployment MP

NOTE

If you have an Operations Manager 1807 or 2019 management group connected to a Log Analytics workspace with agents configured in the management group to collect log data, you need to override the parameter `IsAutoRegistrationEnabled` and set it to `True` in the `Microsoft.IntelligencePacks.AzureAutomation.HybridAgent.Init` rule.

For more information about updates to management packs, see [Connect Operations Manager to Azure Monitor logs](#).

NOTE

For Update Management to fully manage machines with the Log Analytics agent, you must update to the Log Analytics agent for Windows or the Log Analytics agent for Linux. To learn how to update the agent, see [How to upgrade an Operations Manager agent](#). In environments that use Operations Manager, you must be running System Center Operations Manager 2012 R2 UR 14 or later.

Data collection frequency

Update Management scans managed machines for data using the following rules. It can take between 30 minutes and 6 hours for the dashboard to display updated data from managed machines.

- Each Windows machine - Update Management does a scan twice per day for each machine.
- Each Linux machine - Update Management does a scan every hour.

The average data usage by Azure Monitor logs for a machine using Update Management is approximately 25 MB per month. This value is only an approximation and is subject to change, depending on your environment. We recommend that you monitor your environment to keep track of your exact usage. For more information about analyzing Azure Monitor Logs data usage, see [Manage usage and cost](#).

Update classifications

The following table defines the classifications that Update Management supports for Windows updates.

CLASSIFICATION	DESCRIPTION
Critical updates	An update for a specific problem that addresses a critical, non-security-related bug.
Security updates	An update for a product-specific, security-related issue.
Update rollups	A cumulative set of hotfixes that are packaged together for easy deployment.

CLASSIFICATION	DESCRIPTION
Feature packs	New product features that are distributed outside a product release.
Service packs	A cumulative set of hotfixes that are applied to an application.
Definition updates	An update to virus or other definition files.
Tools	A utility or feature that helps complete one or more tasks.
Updates	An update to an application or file that currently is installed.

The next table defines the supported classifications for Linux updates.

CLASSIFICATION	DESCRIPTION
Critical and security updates	Updates for a specific problem or a product-specific, security-related issue.
Other updates	All other updates that aren't critical in nature or that aren't security updates.

NOTE

Update classification for Linux machines is only available when used in supported Azure public cloud regions. There is no classification of Linux updates when using Update Management in the following national cloud regions:

- Azure US Government
- 21Vianet in China

Instead of being classified, updates are reported under the **Other updates** category.

Update Management uses data published by the supported distributions, specifically their released **OVAL** (Open Vulnerability and Assessment Language) files. Because internet access is restricted from these national clouds, Update Management cannot access the files.

For Linux, Update Management can distinguish between critical updates and security updates in the cloud under classification **Security** and **Others**, while displaying assessment data due to data enrichment in the cloud. For patching, Update Management relies on classification data available on the machine. Unlike other distributions, CentOS does not have this information available in the RTM version. If you have CentOS machines configured to return security data for the following command, Update Management can patch based on classifications.

```
sudo yum -q --security check-update
```

There's currently no supported method to enable native classification-data availability on CentOS. At this time, limited support is provided to customers who might have enabled this feature on their own.

To classify updates on Red Hat Enterprise version 6, you need to install the yum-security plugin. On Red Hat Enterprise Linux 7, the plugin is already a part of yum itself and there's no need to install anything. For more information, see the following Red Hat [knowledge article](#).

When you schedule an update to run on a Linux machine, that for example is configured to install only updates matching the **Security** classification, the updates installed might be different from, or are a subset of, the

updates matching this classification. When an assessment of OS updates pending for your Linux machine is performed, [Open Vulnerability and Assessment Language](#) (OVAL) files provided by the Linux distro vendor is used by Update Management for classification.

Categorization is done for Linux updates as **Security** or **Others** based on the OVAL files, which includes updates addressing security issues or vulnerabilities. But when the update schedule is run, it executes on the Linux machine using the appropriate package manager like YUM, APT, or ZYPPER to install them. The package manager for the Linux distro may have a different mechanism to classify updates, where the results may differ from the ones obtained from OVAL files by Update Management. To manually check the machine and understand which updates are security relevant by your package manager, see [Troubleshoot Linux update deployment](#).

NOTE

Deploying updates by update classification may not work correctly for Linux distros supported by Update Management. This is a result of an issue identified with the naming schema of the OVAL file and this prevents Update Management from properly matching classifications based on filtering rules. Because of the different logic used in security update assessments, results may differ from the security updates applied during deployment. If you have classification set as **Critical** and **Security**, the update deployment will work as expected. Only the *classification of updates* during an assessment is affected.

Update Management for Windows Server machines is unaffected; update classification and deployments are unchanged.

Integrate Update Management with Configuration Manager

Customers who have invested in Microsoft Endpoint Configuration Manager for managing PCs, servers, and mobile devices also rely on the strength and maturity of Configuration Manager to help manage software updates. To learn how to integrate Update Management with Configuration Manager, see [Integrate Update Management with Windows Endpoint Configuration Manager](#).

Third-party updates on Windows

Update Management relies on the locally configured update repository to update supported Windows systems, either WSUS or Windows Update. Tools such as [System Center Updates Publisher](#) allow you to import and publish custom updates with WSUS. This scenario allows Update Management to update machines that use Configuration Manager as their update repository with third-party software. To learn how to configure Updates Publisher, see [Install Updates Publisher](#).

Next steps

- Before enabling and using Update Management, review [Plan your Update Management deployment](#).
- Review commonly asked questions about Update Management in the [Azure Automation frequently asked questions](#).

Supported regions for linked Log Analytics workspace

4/2/2021 • 3 minutes to read • [Edit Online](#)

In Azure Automation, you can enable the Update Management, Change Tracking and Inventory, and Start/Stop VMs during off-hours features for your servers and virtual machines. These features have a dependency on a Log Analytics workspace, and therefore require linking the workspace with an Automation account. However, only certain regions are supported to link them together. In general, the mapping is *not* applicable if you plan to link an Automation account to a workspace that won't have these features enabled.

The mappings discussed here apply only to linking the Log Analytics Workspace to an Automation account. They do not apply to the virtual machines (VMs) that are connected to the workspace that's linked to the Automation Account. VMs aren't limited to the regions supported by a given Log Analytics workspace. They can be in any region. Keep in mind that having the VMs in a different region may affect state, local, and country regulatory requirements, or your company's compliance requirements. Having VMs in a different region could also introduce data bandwidth charges.

Before connecting VMs to a workspace in a different region, you should review the requirements and potential costs to confirm and understand the legal and cost implications.

This article provides the supported mappings in order to successfully enable and use these features in your Automation account.

For more information, see [Log Analytics workspace and Automation account](#).

Supported mappings

NOTE

As shown in following table, only one mapping can exist between Log Analytics and Azure Automation.

The following table shows the supported mappings:

LOG ANALYTICS WORKSPACE REGION	AZURE AUTOMATION REGION
US	
EastUS ¹	EastUS2
EastUS ²	EastUS
WestUS	WestUS
WestUS2	WestUS2
NorthCentralUS	NorthCentralUS
CentralUS	CentralUS

LOG ANALYTICS WORKSPACE REGION	AZURE AUTOMATION REGION
SouthCentralUS	SouthCentralUS
WestCentralUS	WestCentralUS
Brazil	
BrazilSouth	BrazilSouth
Canada	
CanadaCentral	CanadaCentral
China	
ChinaEast2 ³	ChinaEast2
Asia Pacific	
EastAsia	EastAsia
SoutheastAsia	SoutheastAsia
India	
CentralIndia	CentralIndia
Japan	
JapanEast	JapanEast
Australia	
AustraliaEast	AustraliaEast
AustraliaSoutheast	AustraliaSoutheast
Korea	
KoreaCentral	KoreaCentral
Norway	
NorwayEast	NorwayEast
Europe	
NorthEurope	NorthEurope
WestEurope	WestEurope

LOG ANALYTICS WORKSPACE REGION	AZURE AUTOMATION REGION
France	
FranceCentral	FranceCentral
United Kingdom	
UKSouth	UKSouth
Switzerland	
SwitzerlandNorth	SwitzerlandNorth
United Arab Emirates	
UAENorth	UAENorth
US Gov	
USGovVirginia	USGovVirginia
USGovArizona ³	USGovArizona

¹ EastUS mapping for Log Analytics workspaces to Automation accounts isn't an exact region-to-region mapping, but is the correct mapping.

² EastUS2 mapping for Log Analytics workspaces to Automation accounts isn't an exact region-to-region mapping, but is the correct mapping.

³ In this region, only Update Management is supported, and other features like Change Tracking and Inventory are not available at this time.

Unlink a workspace

If you decide that you no longer want to integrate your Automation account with a Log Analytics workspace, you can unlink your account directly from the Azure portal. Before proceeding, you first need to [remove](#) Update Management, Change Tracking and Inventory, and Start/Stop VMs during off-hours if you are using them. If you don't remove them, you can't complete the unlinking operation.

With the features removed, you can follow the steps below to unlink your Automation account.

NOTE

Some features, including earlier versions of the Azure SQL monitoring solution, might have created Automation assets that need to be removed prior to unlinking the workspace.

- From the Azure portal, open your Automation account. On the Automation account page, select **Linked workspace** under **Related Resources**.
- On the Unlink workspace page, select **Unlink workspace**. You receive a prompt verifying if you want to continue.
- While Azure Automation is unlinking the account from your Log Analytics workspace, you can track the

progress under **Notifications** from the menu.

4. If you used Update Management, optionally you might want to remove the following items that are no longer needed:
 - Update schedules: Each has a name that matches an update deployment that you created.
 - Hybrid worker groups created for the feature: Each has a name similar to
`machine1.contoso.com_9ceb8108-26c9-4051-b6b3-227600d715c8`.
5. If you used Start/Stop VMs during off-hours, optionally you can remove the following items that are no longer needed:
 - Start and stop VM runbook schedules
 - Start and stop VM runbooks
 - Variables

Alternatively, you can unlink your workspace from your Automation account within the workspace.

1. In the workspace, select **Automation Account** under **Related Resources**.
2. On the Automation Account page, select **Unlink account**.

Next steps

- Learn about Update Management in [Update Management overview](#).
- Learn about Change Tracking and Inventory in [Change Tracking and Inventory overview](#).
- Learn about Start/Stop VMs during off-hours in [Start/Stop VMs during off-hours overview](#).

Plan your Update Management deployment

8/24/2021 • 5 minutes to read • [Edit Online](#)

Step 1 - Automation account

Update Management is an Azure Automation feature, and therefore requires an Automation account. You can use an existing Automation account in your subscription, or create a new account dedicated only for Update Management and no other Automation features.

Step 2 - Azure Monitor Logs

Update Management depends on a Log Analytics workspace in Azure Monitor to store assessment and update status log data collected from managed machines. Integration with Log Analytics also enables detailed analysis and alerting in Azure Monitor. You can use an existing workspace in your subscription, or create a new one dedicated only for Update Management.

If you are new to Azure Monitor Logs and the Log Analytics workspace, you should review the [Design a Log Analytics workspace](#) deployment guide.

Step 3 - Supported operating systems

Update Management supports specific versions of the Windows Server and Linux operating systems. Before you enable Update Management, confirm that the target machines meet the [operating system requirements](#).

Step 4 - Log Analytics agent

The [Log Analytics agent](#) for Windows and Linux is required to support Update Management. The agent is used for both data collection, and the Automation system Hybrid Runbook Worker role to support Update Management runbooks used to manage the assessment and update deployments on the machine.

On Azure VMs, if the Log Analytics agent isn't already installed, when you enable Update Management for the VM it is automatically installed using the Log Analytics VM extension for [Windows](#) or [Linux](#). The agent is configured to report to the Log Analytics workspace linked to the Automation account Update Management is enabled in.

Non-Azure VMs or servers need to have the Log Analytics agent for Windows or Linux installed and reporting to the linked workspace. We recommend installing the Log Analytics agent for Windows or Linux by first connecting your machine to [Azure Arc-enabled servers](#), and then use Azure Policy to assign the [Deploy Log Analytics agent to Linux or Windows Azure Arc machines](#) built-in policy definition. Alternatively, if you plan to monitor the machines with [VM insights](#), instead use the [Enable Azure Monitor for VMs](#) initiative.

If you're enabling a machine that's currently managed by Operations Manager, a new agent isn't required. The workspace information is added to the agents configuration when you connect the management group to the Log Analytics workspace.

Having a machine registered for Update Management in more than one Log Analytics workspace (also referred to as multihoming) isn't supported.

Step 5 - Network planning

To prepare your network to support Update Management, you may need to configure some infrastructure components. For example, open firewall ports to pass the communications used by Update Management and

Azure Monitor.

Review [Azure Automation Network Configuration](#) for detailed information on the ports, URLs, and other networking details required for Update Management, including the Hybrid Runbook Worker role. To connect to the Automation service from your Azure VMs securely and privately, review [Use Azure Private Link](#).

For Windows machines, you must also allow traffic to any endpoints required by Windows Update agent. You can find an updated list of required endpoints in [Issues related to HTTP/Proxy](#). If you have a local [Windows Server Update Services](#) (WSUS) deployment, you must also allow traffic to the server specified in your [WSUS key](#).

For Red Hat Linux machines, see [IPs for the RHUI content delivery servers](#) for required endpoints. For other Linux distributions, see your provider documentation.

If your IT security policies do not allow machines on the network to connect to the internet, you can set up a [Log Analytics gateway](#) and then configure the machine to connect through the gateway to Azure Automation and Azure Monitor.

Step 6 - Permissions

To create and manage update deployments, you need specific permissions. To learn about these permissions, see [Role-based access - Update Management](#).

Step 7 - Windows Update client

Azure Automation Update Management relies on the Windows Update client to download and install Windows updates. There are specific group policy settings that are used by Windows Update Agent (WUA) on machines to connect to Windows Server Update Services (WSUS) or Microsoft Update. These group policy settings are also used to successfully scan for software update compliance, and to automatically update the software updates. To review our recommendations, see [Configure Windows Update settings for Update Management](#).

Step 8 - Linux repository

VMs created from the on-demand Red Hat Enterprise Linux (RHEL) images available in Azure Marketplace are registered to access the Red Hat Update Infrastructure (RHUI) that's deployed in Azure. Any other Linux distribution must be updated from the distribution's online file repository by using methods supported by that distribution.

To classify updates on Red Hat Enterprise version 6, you need to install the yum-security plugin. On Red Hat Enterprise Linux 7, the plugin is already a part of yum itself and there's no need to install anything. For more information, see the following Red Hat [knowledge article](#).

Step 9 - Plan deployment targets

Update Management allows you to target updates to a dynamic group representing Azure or non-Azure machines, so you can ensure that specific machines always get the right updates at the most convenient times. A dynamic group is resolved at deployment time and is based on the following criteria:

- Subscription
- Resource groups
- Locations
- Tags

For non-Azure machines, a dynamic group uses saved searches, also called [computer groups](#). Update deployments scoped to a group of machines is only visible from the Automation account in the Update

Management **Deployment schedules** option, not from a specific Azure VM.

Alternatively, updates can be managed only for a selected Azure VM. Update deployments scoped to the specific machine are visible from both the machine and from the Automation account in Update Management **Deployment schedules** option.

Next steps

Enable Update Management and select machines to be managed using one of the following methods:

- Using an Azure [Resource Manager template](#) to deploy Update Management to a new or existing Automation account and Azure Monitor Log Analytics workspace in your subscription. It does not configure the scope of machines that should be managed, this is performed as a separate step after using the template.
- From your [Automation account](#) for one or more Azure and non-Azure machines, including Arc-enabled servers.
- Using the [Enable-AutomationSolution runbook](#) to automate onboarding Azure VMs.
- For a [selected Azure VM](#) from the [Virtual machines](#) page in the Azure portal. This scenario is available for Linux and Windows VMs.
- For [multiple Azure VMs](#) by selecting them from the [Virtual machines](#) page in the Azure portal.

Operating systems supported by Update Management

8/24/2021 • 4 minutes to read • [Edit Online](#)

This article details the Windows and Linux operating systems supported and system requirements for machines or servers managed by Update Management.

Supported operating systems

The following table lists the supported operating systems for update assessments and patching. Patching requires a system Hybrid Runbook Worker, which is automatically installed when you enable the virtual machine or server for management by Update Management. For information on Hybrid Runbook Worker system requirements, see [Deploy a Windows Hybrid Runbook Worker](#) and [Deploy a Linux Hybrid Runbook Worker](#).

All operating systems are assumed to be x64. x86 is not supported for any operating system.

NOTE

Update assessment of Linux machines is only supported in certain regions as listed in the Automation account and Log Analytics workspace [mappings table](#).

OPERATING SYSTEM	NOTES
Windows Server 2019 (Datacenter/Standard including Server Core)	
Windows Server 2016 (Datacenter/Standard excluding Server Core)	
Windows Server 2012 R2(Datacenter/Standard)	
Windows Server 2012	
Windows Server 2008 R2 (RTM and SP1 Standard)	Update Management supports assessments and patching for this operating system. The Hybrid Runbook Worker is supported for Windows Server 2008 R2.
CentOS 6, 7, and 8	Linux agents require access to an update repository. Classification-based patching requires <code>yum</code> to return security data that CentOS doesn't have in its RTM releases. For more information on classification-based patching on CentOS, see Update classifications on Linux .
Oracle Linux 6.x, 7.x, 8x	Linux agents require access to an update repository.
Red Hat Enterprise 6, 7, and 8	Linux agents require access to an update repository.
SUSE Linux Enterprise Server 12, 15, 15.1, and 15.2	Linux agents require access to an update repository.

OPERATING SYSTEM	NOTES
Ubuntu 14.04 LTS, 16.04 LTS, 18.04 LTS, and 20.04 LTS	Linux agents require access to an update repository.

NOTE

Update Management does not support safely automating update management across all instances in an Azure virtual machine scale set. [Automatic OS image upgrades](#) is the recommended method for managing OS image upgrades on your scale set.

Unsupported operating systems

The following table lists operating systems not supported by Update Management:

OPERATING SYSTEM	NOTES
Windows client	Client operating systems (such as Windows 7 and Windows 10) aren't supported. For Azure Windows Virtual Desktop (WVD), the recommended method to manage updates is Microsoft Endpoint Configuration Manager for Windows 10 client machine patch management.
Windows Server 2016 Nano Server	Not supported.
Azure Kubernetes Service Nodes	Not supported. Use the patching process described in Apply security and kernel updates to Linux nodes in Azure Kubernetes Service (AKS)

System requirements

The following information describes operating system-specific requirements. For additional guidance, see [Network planning](#). To understand requirements for TLS 1.2, see [TLS 1.2 for Azure Automation](#).

Windows

Software Requirements:

- .NET Framework 4.6 or later is required. ([Download the .NET Framework](#).)
- Windows PowerShell 5.1 is required ([Download Windows Management Framework 5.1](#).)
- The Update Management feature depends on the system Hybrid Runbook Worker role, and you should confirm its [system requirements](#).

Windows Update agents must be configured to communicate with a Windows Server Update Services (WSUS) server, or they require access to Microsoft Update. For hybrid machines, we recommend installing the Log Analytics agent for Windows by first connecting your machine to [Azure Arc-enabled servers](#), and then use Azure Policy to assign the [Deploy Log Analytics agent to Windows Azure Arc machines](#) built-in policy definition. Alternatively, if you plan to monitor the machines with VM insights, instead use the [Enable VM insights](#) initiative.

You can use Update Management with Microsoft Endpoint Configuration Manager. To learn more about integration scenarios, see [Integrate Update Management with Windows Endpoint Configuration Manager](#). The [Log Analytics agent for Windows](#) is required for Windows servers managed by sites in your Configuration Manager environment.

By default, Windows VMs that are deployed from Azure Marketplace are set to receive automatic updates from Windows Update Service. This behavior doesn't change when you add Windows VMs to your workspace. If you don't actively manage updates by using Update Management, the default behavior (to automatically apply updates) applies.

NOTE

You can modify Group Policy so that machine reboots can be performed only by the user, not by the system. Managed machines can get stuck if Update Management doesn't have rights to reboot the machine without manual interaction from the user. For more information, see [Configure Group Policy settings for Automatic Updates](#).

Linux

Software Requirements:

- The machine requires access to an update repository, either private or public.
- TLS 1.1 or TLS 1.2 is required to interact with Update Management.
- The Update Management feature depends on the system Hybrid Runbook Worker role, and you should confirm its [system requirements](#). Because Update Management uses Automation runbooks to initiate assessment and update of your machines, review the [version of Python required](#) for your supported Linux distro.

NOTE

Update assessment of Linux machines is only supported in certain regions. See the Automation account and Log Analytics workspace [mappings table](#).

For hybrid machines, we recommend installing the Log Analytics agent for Linux by first connecting your machine to [Azure Arc-enabled servers](#), and then use Azure Policy to assign the [Deploy Log Analytics agent to Linux Azure Arc machines](#) built-in policy definition. Alternatively, if you plan to monitor the machines with Azure Monitor for VMs, instead use the [Enable Azure Monitor for VMs](#) initiative.

Next steps

Before you enable and use Update Management, review [Plan your Update Management deployment](#).

Enable Update Management using Azure Resource Manager template

7/23/2021 • 6 minutes to read • [Edit Online](#)

You can use an [Azure Resource Manager template](#) to enable the Azure Automation Update Management feature in your resource group. This article provides a sample template that automates the following:

- Automates the creation of an Azure Monitor Log Analytics workspace.
- Automates the creation of an Azure Automation account.
- Links the Automation account to the Log Analytics workspace.
- Adds sample Automation runbooks to the account.
- Enables the Update Management feature.

If you already have a Log Analytics workspace and Automation account deployed in a supported region in your subscription, they are not linked. Using this template successfully creates the link and deploys Update Management.

NOTE

Creation of the Automation Run As account is not supported when you're using an ARM template. To create a Run As account manually from the portal or with PowerShell, see [Create Run As account](#).

After you complete these steps, you need to [configure diagnostic settings](#) for your Automation account to send runbook job status and job streams to the linked Log Analytics workspace.

API versions

The following table lists the API version for the resources used in this example.

RESOURCE	RESOURCE TYPE	API VERSION
Workspace	workspaces	2020-03-01-preview
Automation account	automation	2020-01-13-preview
Workspace Linked services	workspaces	2020-03-01-preview
Solutions	solutions	2015-11-01-preview

Before using the template

The JSON template is configured to prompt you for:

- The name of the workspace.
- The region to create the workspace in.
- The name of the Automation account.
- The region to create the Automation account in.

The following parameters in the template are set with a default value for the Log Analytics workspace:

- *sku* defaults to the per GB pricing tier released in the April 2018 pricing model.
- *dataRetention* defaults to 30 days.

WARNING

If you want to create or configure a Log Analytics workspace in a subscription that has opted into the April 2018 pricing model, the only valid Log Analytics pricing tier is *PerGB2018*.

The JSON template specifies a default value for the other parameters that would likely be used as a standard configuration in your environment. You can store the template in an Azure storage account for shared access in your organization. For more information about working with templates, see [Deploy resources with ARM templates and the Azure CLI](#).

If you're new to Azure Automation and Azure Monitor, it's important that you understand the following configuration details. They can help you avoid errors when you try to create, configure, and use a Log Analytics workspace linked to your new Automation account.

- Review [additional details](#) to fully understand workspace configuration options, such as access control mode, pricing tier, retention, and capacity reservation level.
- Review [workspace mappings](#) to specify the supported regions inline or in a parameter file. Only certain regions are supported for linking a Log Analytics workspace and an Automation account in your subscription.
- If you're new to Azure Monitor logs and have not deployed a workspace already, you should review the [workspace design guidance](#). It will help you to learn about access control, and understand the design implementation strategies we recommend for your organization.

Deploy template

1. Copy and paste the following JSON syntax into your file:

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "workspaceName": {
      "type": "string",
      "metadata": {
        "description": "Workspace name"
      }
    },
    "sku": {
      "type": "string",
      "defaultValue": "pergb2018",
      "allowedValues": [
        "pergb2018",
        "Free",
        "Standalone",
        "PerNode",
        "Standard",
        "Premium"
      ],
      "metadata": {
        "description": "Pricing tier: perGB2018 or legacy tiers (Free, Standalone, PerNode, Standard or Premium), which are not available to all customers."
      }
    },
    "dataRetention": {
      "type": "int".
    }
  }
}
```

```
        },
        "defaultValue": 30,
        "minValue": 7,
        "maxValue": 730,
        "metadata": {
            "description": "Number of days to retain data."
        }
    },
    "location": {
        "type": "string",
        "defaultValue": "[resourceGroup().location]",
        "metadata": {
            "description": "Specifies the location in which to create the workspace."
        }
    },
    "automationAccountName": {
        "type": "string",
        "metadata": {
            "description": "Automation account name"
        }
    },
    "sampleGraphicalRunbookName": {
        "type": "String",
        "defaultValue": "AzureAutomationTutorial"
    },
    "sampleGraphicalRunbookDescription": {
        "type": "String",
        "defaultValue": "An example runbook that gets all the Resource Manager resources by using the Run As account (service principal)."
    },
    "samplePowerShellRunbookName": {
        "type": "String",
        "defaultValue": "AzureAutomationTutorialScript"
    },
    "samplePowerShellRunbookDescription": {
        "type": "String",
        "defaultValue": "An example runbook that gets all the Resource Manager resources by using the Run As account (service principal)."
    },
    "samplePython2RunbookName": {
        "type": "String",
        "defaultValue": "AzureAutomationTutorialPython2"
    },
    "samplePython2RunbookDescription": {
        "type": "String",
        "defaultValue": "An example runbook that gets all the Resource Manager resources by using the Run As account (service principal)."
    },
    "_artifactsLocation": {
        "type": "string",
        "defaultValue": "[deployment().properties.templateLink.uri]",
        "metadata": {
            "description": "URI to artifacts location"
        }
    },
    "_artifactsLocationSasToken": {
        "type": "securestring",
        "defaultValue": "",
        "metadata": {
            "description": "The sasToken required to access _artifactsLocation. When the template is deployed using the accompanying scripts, a sasToken will be automatically generated"
        }
    },
    "variables": {
        "Updates": {
            "name": "[concat('Updates', '(', parameters('workspaceName'), ')')]",
            "galleryName": "Updates"
        }
    }
}
```

```
j,
"resources": [
{
  "type": "Microsoft.OperationalInsights/workspaces",
  "apiVersion": "2020-08-01",
  "name": "[parameters('workspaceName')]",
  "location": "[parameters('location')]",
  "properties": {
    "sku": {
      "name": "[parameters('sku')]"
    },
    "retentionInDays": "[parameters('dataRetention')]",
    "features": {
      "searchVersion": 1,
      "legacy": 0
    }
  }
},
{
  "apiVersion": "2015-11-01-preview",
  "location": "[parameters('location')]",
  "name": "[variables('Updates').name]",
  "type": "Microsoft.OperationsManagement/solutions",
  "id": "[concat('/subscriptions/', subscription().subscriptionId, '/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.OperationsManagement/solutions/',
variables('Updates').name)]",
  "dependsOn": [
    "[concat('Microsoft.OperationalInsights/workspaces/', parameters('workspaceName'))]"
  ],
  "properties": {
    "workspaceResourceId": "[resourceId('Microsoft.OperationalInsights/workspaces/',
parameters('workspaceName'))]"
  },
  "plan": {
    "name": "[variables('Updates').name]",
    "publisher": "Microsoft",
    "promotionCode": "",
    "product": "[concat('OMSGallery/', variables('Updates').galleryName)]"
  }
},
{
  "type": "Microsoft.Automation/automationAccounts",
  "apiVersion": "2020-01-13-preview",
  "name": "[parameters('automationAccountName')]",
  "location": "[parameters('location')]",
  "dependsOn": [
    "[parameters('workspaceName')]"
  ],
  "identity": {
    "type": "SystemAssigned"
  },
  "properties": {
    "sku": {
      "name": "Basic"
    }
  },
  "resources": [
{
  "type": "runbooks",
  "apiVersion": "2020-01-13-preview",
  "name": "[parameters('sampleGraphicalRunbookName')]",
  "location": "[parameters('location')]",
  "dependsOn": [
    "[parameters('automationAccountName')]"
  ],
  "properties": {
    "runbookType": "GraphPowerShell",
    "logProgress": "false",
    "logVerbose": "false",
    "scriptContent": "[redacted]"
  }
}
]
```

```

    "description": "[parameters('sampleGraphicalRunbookDescription')]",
    "publishContentLink": {
        "uri": "[uri(parameters('_artifactsLocation'),
concat('scripts/AzureAutomationTutorial.graphrunbook', parameters('_artifactsLocationSasToken')))]",
        "version": "1.0.0.0"
    }
},
{
    "type": "runbooks",
    "apiVersion": "2020-01-13-preview",
    "name": "[parameters('samplePowerShellRunbookName')]",
    "location": "[parameters('location')]",
    "dependsOn": [
        "[parameters('automationAccountName')]"
    ],
    "properties": {
        "runbookType": "PowerShell",
        "logProgress": "false",
        "logVerbose": "false",
        "description": "[parameters('samplePowerShellRunbookDescription')]",
        "publishContentLink": {
            "uri": "[uri(parameters('_artifactsLocation'),
concat('scripts/AzureAutomationTutorial.ps1', parameters('_artifactsLocationSasToken')))]",
            "version": "1.0.0.0"
        }
    }
},
{
    "type": "runbooks",
    "apiVersion": "2020-01-13-preview",
    "name": "[parameters('samplePython2RunbookName')]",
    "location": "[parameters('location')]",
    "dependsOn": [
        "[parameters('automationAccountName')]"
    ],
    "properties": {
        "runbookType": "Python2",
        "logProgress": "false",
        "logVerbose": "false",
        "description": "[parameters('samplePython2RunbookDescription')]",
        "publishContentLink": {
            "uri": "[uri(parameters('_artifactsLocation'),
concat('scripts/AzureAutomationTutorialPython2.py', parameters('_artifactsLocationSasToken')))]",
            "version": "1.0.0.0"
        }
    }
},
{
    "type": "Microsoft.OperationalInsights/workspaces/linkedServices",
    "apiVersion": "2020-08-01",
    "name": "[concat(parameters('workspaceName'), '/', 'Automation')]",
    "location": "[parameters('location')]",
    "dependsOn": [
        "[parameters('workspaceName')]",
        "[parameters('automationAccountName')]"
    ],
    "properties": {
        "resourceId": "[resourceId('Microsoft.Automation/automationAccounts',
parameters('automationAccountName'))]"
    }
}
]
}

```

2. Edit the template to meet your requirements. Consider creating a [Resource Manager parameters file](#)

instead of passing parameters as inline values.

3. Save this file to a local folder as **deployUMSolutiontemplate.json**.
4. You are ready to deploy this template. You can use either PowerShell or the Azure CLI. When you're prompted for a workspace and Automation account name, provide a name that is globally unique across all Azure subscriptions.

PowerShell

```
New-AzResourceGroupDeployment ` 
    -Name <deployment-name> ` 
    -ResourceGroupName <resource-group-name> ` 
    -TemplateFile deployUMSolutiontemplate.json ` 
    -_artifactsLocation "https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/quickstarts/microsoft.automation/101-automation/azuredetect.json"
```

Azure CLI

```
az deployment group create --resource-group <my-resource-group> --name <my-deployment-name> -- 
    template-file deployUMSolutiontemplate.json --parameters 
    _artifactsLocation="https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/quickstarts/microsoft.automation/101-automation/azuredetect.json"
```

The deployment can take a few minutes to complete. When it finishes, you see a message similar to the following that includes the result:

DeploymentName	:	DeployAALAAU
ResourceGroupName	:	Prod1
ProvisioningState	:	Succeeded
Timestamp	:	9/18/2020 6:51:42 PM
Mode	:	Incremental
TemplateLink	:	
Parameters	:	
	:	
Name	Type	Value
workspaceName	String	Prod1-LA
sku	String	pergb2018
dataRetention	Int	30
location	String	eastus
automationAccountName	String	Prod1-AA
automationAccountLocation	String	eastus2
sampleGraphicalRunbookName	String	AzureAutomationTutorial
sampleGraphicalRunbookDescription	String	An example runbook that gets all the Resource Manager resources by using the Run As account (service principal).
samplePowerShellRunbookName	String	AzureAutomationTutorialScript
samplePowerShellRunbookDescription	String	An example runbook that gets all the Resource Manager resources by using the Run As account (service principal).
samplePython2RunbookName	String	AzureAutomationTutorialPython2
samplePython2RunbookDescription	String	An example runbook that gets all the Resource Manager resources by using the Run As account (service principal).
_artifactsLocation	String	https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/101-automation/
_artifactsLocationSasToken	SecureString	

Review deployed resources

1. Sign in to the [Azure portal](#).
2. In the Azure portal, open the Automation account you created.
3. From the left-pane, select **Runbooks**. On the **Runbooks** page, listed are three tutorial runbooks created with the Automation account.

Name	Authoring status	Runbook type	Last modified
AzureAutomationTutorial	✓ Published	Graphical Runbook	7/23/2020, 10:29 AM
AzureAutomationTutorialPython...	✓ Published	Python 2 Runbook	7/23/2020, 10:29 AM
AzureAutomationTutorialScript	✓ Published	PowerShell Runbook	7/23/2020, 10:29 AM

4. From the left-pane, select **Linked workspace**. On the **Linked workspace** page, it shows the Log Analytics workspace you specified earlier linked to your Automation account.

Prod1-AA | Linked workspace
Automation Account

Search (Ctrl+ /) Go to workspace Unlink workspace

- Modules
- Modules gallery
- Python 2 packages
- Credentials
- Connections
- Certificates
- Variables

Related Resources

- Linked workspace

This Automation account is linked to the following Log Analytics workspace: [prod1-la](#)

Unlink workspace

To unlink this Automation account and the Log Analytics workspace you must first remove some solutions. These are the following:

- Update Management
- Change Tracking
- Start/Stop VMs during off-hours

After you remove these solutions you can click **Unlink workspace** above to complete the unlinking process.

If you use the Update Management solution you optionally may want to remove some items.

5. From the left-pane, select **Update management**. On the **Update management** page, it shows the assessment page without any information as a result of just being enabled, and machines aren't configured for management.

Dashboard > Automation Accounts > MAIC-AA-Pri

Prod1-AA-Pri | Update management Automation Account

Schedule update deployment Add Azure VMs Add non-Azure machine Manage machines

Non-compliant machines	Machines need attention (0)	Missing updates (0)
0 ✓ out of 0	Critical and security 0 Other 0 Not assessed 0	Critical 0 Security 0 Others 0

Machines (0) Missing updates (0) Deployment schedules History

Filter by name Compliance: All Platform

Machine name	Compliance	Platform	Operating system	Critical
No machines currently appear assessed. For the machines connected just recently it might take a few minutes to start showing up.				

Clean up resources

When you no longer need them, delete the **Updates** solution in the Log Analytics workspace, unlink the Automation account from the workspace, and then delete the Automation account and workspace.

Next steps

- To use Update Management for VMs, see [Manage updates and patches for your VMs](#).
- If you no longer want to use Update Management and wish to remove it, see instructions in [Remove Update Management feature](#).
- To delete VMs from Update Management, see [Remove VMs from Update Management](#).

Enable Update Management from the Azure portal

3/5/2021 • 2 minutes to read • [Edit Online](#)

This article describes how you can enable the [Update Management](#) feature for VMs by browsing the Azure portal. To enable Azure VMs at scale, you must enable an existing Azure VM using Update Management.

The number of resource groups that you can use for managing your VMs is limited by the [Resource Manager deployment limits](#). Resource Manager deployments, not to be confused with Update deployments, are limited to five resource groups per deployment. Two of these resource groups are reserved to configure the Log Analytics workspace, Automation account, and related resources. This leaves you with three resource groups to select for management by Update Management. This limit only applies to simultaneous setup, not the number of resource groups that can be managed by an Automation feature.

NOTE

When enabling Update Management, only certain regions are supported for linking a Log Analytics workspace and an Automation Account. For a list of the supported mapping pairs, see [Region mapping for Automation Account and Log Analytics workspace](#).

Prerequisites

- Azure subscription. If you don't have one yet, you can [activate your MSDN subscriber benefits](#) or sign up for a [free account](#).
- [Automation account](#) to manage machines.
- A [virtual machine](#).

Sign in to Azure

Sign in to Azure at <https://portal.azure.com>.

Enable Update Management

1. In the Azure portal, navigate to **Virtual machines**.
2. On the **Virtual machines** page, use the checkboxes to choose the VMs to add to Update Management.
You can add machines for up to three different resource groups at a time. Azure VMs can exist in any region, no matter the location of your Automation account.

Home >

Virtual machines

InfraLabAD

+ Add ⌂ Reservations Edit columns Refresh ⌂ Assign tags ⌂ Start ⌂ Restart ⌂ Stop Delete ⌂ Services

Subscriptions: Parnell Aerospace – Don't see a subscription? [Open Directory + Subscription settings](#)

Filter by name... MAIC-RG All types All locations Change Tracking

9 of 9 items selected

Name	Type	Status	Resource group	Location	Source
CMPRI01	Virtual machine	Stopped (deallocated)	maic-rg	East US 2	Marketplace
DC01	Virtual machine	Running	MAIC-RG	East US 2	Marketplace
DC02	Virtual machine	Running	MAIC-RG	East US 2	Marketplace
OMMS01	Virtual machine	Running	MAIC-RG	East US 2	Marketplace
OMMS02	Virtual machine	Running	MAIC-RG	East US 2	Marketplace
OMSSQLServer01	Virtual machine	Running	maic-rg	East US 2	Marketplace
slessrvr01	Virtual machine	Running	maic-rg	East US 2	Marketplace
SVR01	Virtual machine	Running	MAIC-RG	East US 2	Marketplace
SVR03	Virtual machine	Running	maic-rg	East US 2	Marketplace

TIP

Use the filter controls to select VMs from different subscriptions, locations, and resource groups. You can click the top checkbox to select all virtual machines in a list.

Home > Virtual machines >

Enable Update Management

 **Update Management**

Enable consistent control and compliance of these virtual machines with Update Management.

Subscription: Parnell Aerospace (virtual machines: 9) Location: East US 2 (virtual machines: 9)

Configuration (used when enabling new VMs) [ⓘ](#)

AUTO: Auto-configure Log Analytics workspace and Automation account based on VMs subscription and location

Log Analytics workspace: DefaultWorkspace-68627f8c-65b8-4601-b48e-b032a81f8cf0-EUS
Automation account: Automate-68627f8c-65b8-4601-b48e-b032a81f8cf0-EUS

Summary

Ready to enable	Already enabled	Cannot enable
1 →	5 ✓	3 -

Name [Update Management... ⓘ](#) Details

slessrvr01 → Ready to enable

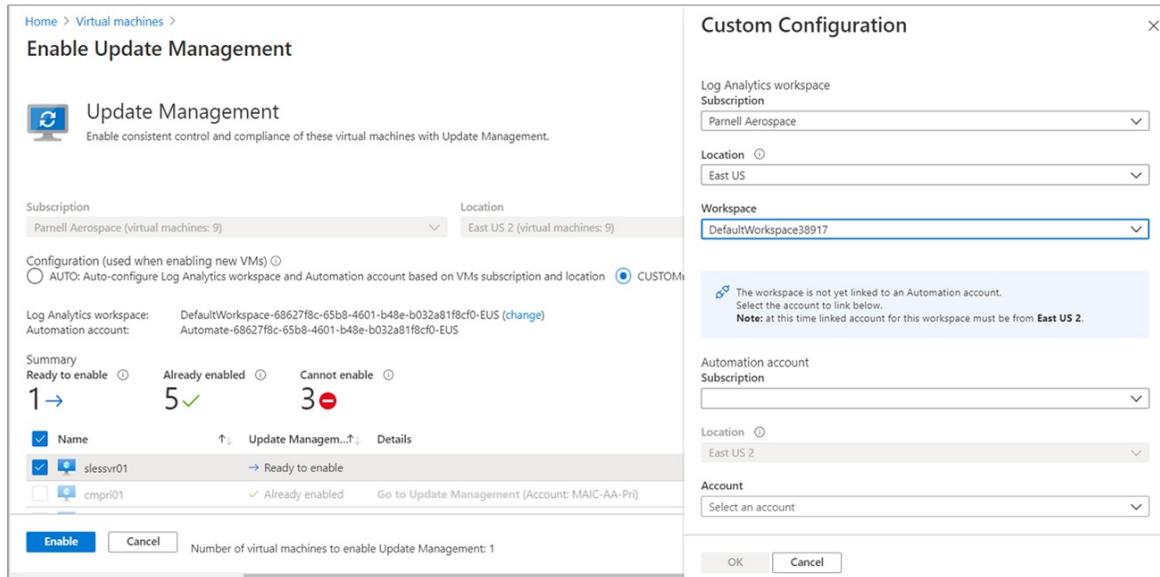
cmpri01 ✓ Already enabled Go to Update Management (Account: MAIC-AA)

Enable **Cancel** Number of virtual machines to enable Update Management: 1

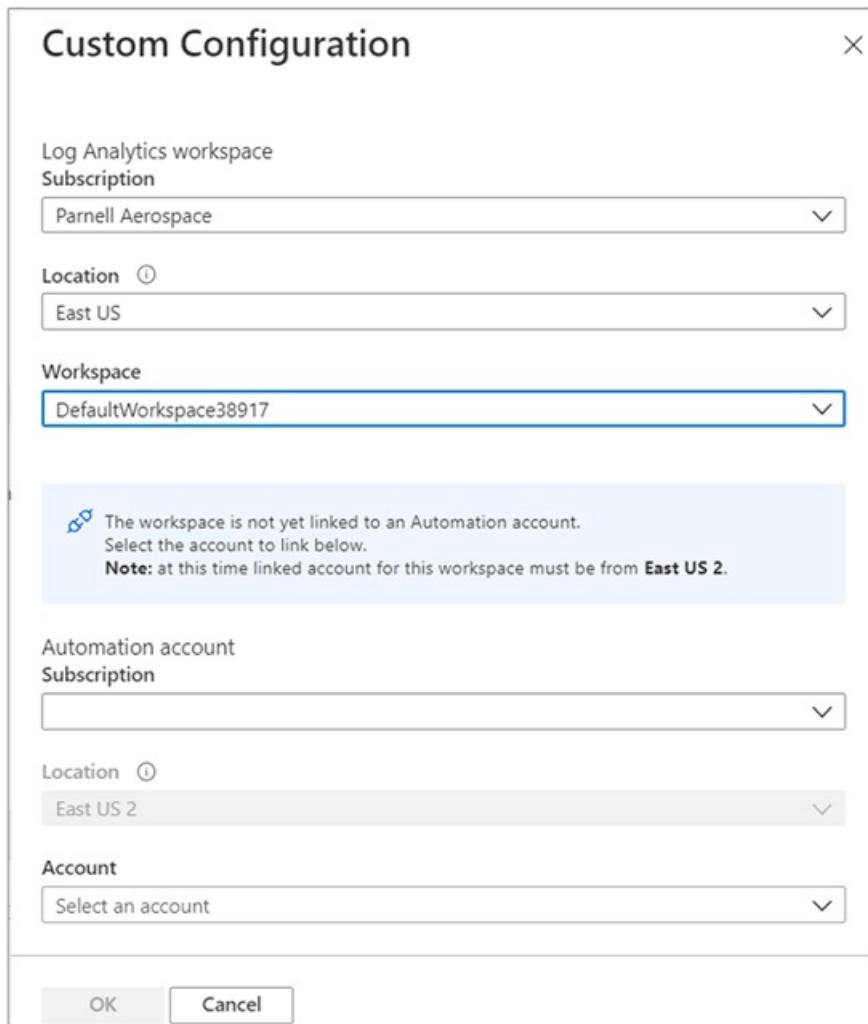
3. Select **Services** and select **Update Management** for the Update Management feature.
4. The list of virtual machines is filtered to show only the virtual machines that are in the same subscription

and location. If your virtual machines are in more than three resource groups, the first three resource groups are selected.

5. An existing Log Analytics workspace and Automation account are selected by default. If you want to use a different Log Analytics workspace and Automation account, select **CUSTOM** to select them from the Custom Configuration page. When you choose a Log Analytics workspace, a check is made to determine if it is linked with an Automation account. If a linked Automation account is found, you see the following screen. When done, select **OK**.



6. If the workspace selected is not linked to an Automation account, you see the following screen. Select an Automation account and select **OK** when finished.



7. Deselect any virtual machine that you don't want to enable. VMs that can't be enabled are already deselected.
8. Select **Enable** to enable the feature. After you've enabled Update Management, it might take about 15 minutes before you can view the update assessment from them.

Next steps

- To use Update Management for VMs, see [Manage updates and patches for your VMs](#).
- To troubleshoot general Update Management errors, see [Troubleshoot Update Management issues](#).
- To troubleshoot problems with the Windows update agent, see [Troubleshoot Windows update agent issues](#).
- To troubleshoot problems with the Linux update agent, see [Troubleshoot Linux update agent issues](#).

Enable Update Management from an Azure VM

3/5/2021 • 2 minutes to read • [Edit Online](#)

This article describes how you can enable the [Update Management](#) feature on one or more Azure virtual machines (VM). To enable Azure VMs at scale, you must enable an existing Azure VM using Update Management.

NOTE

When enabling Update Management, only certain regions are supported for linking a Log Analytics workspace and an Automation account. For a list of the supported mapping pairs, see [Region mapping for Automation account and Log Analytics workspace](#).

Prerequisites

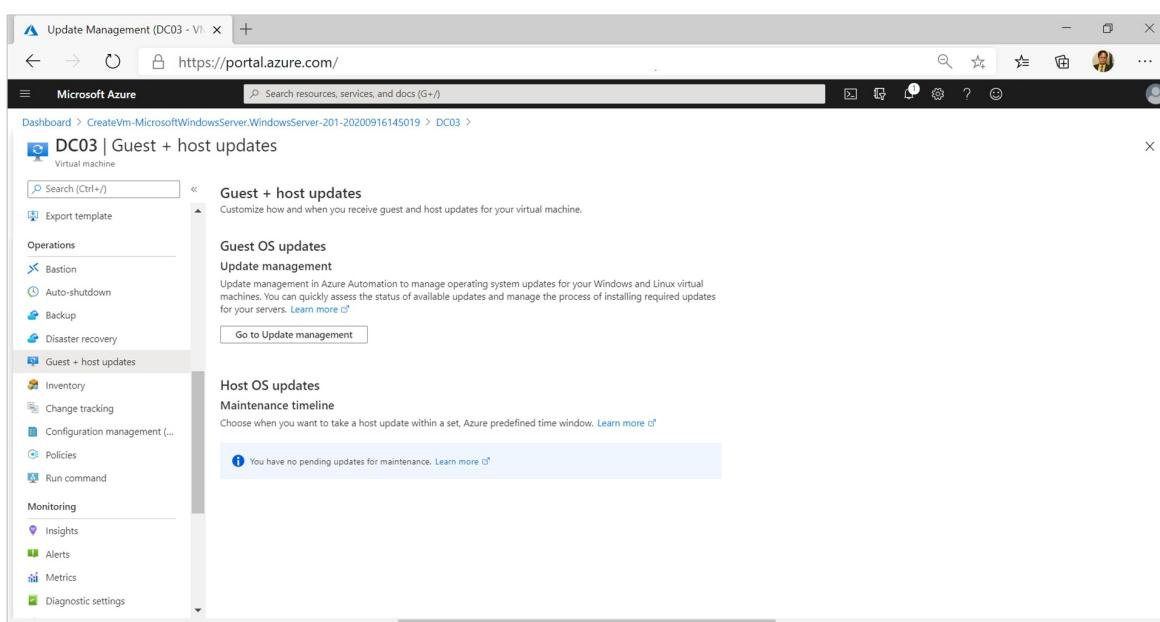
- Azure subscription. If you don't have one yet, you can [activate your MSDN subscriber benefits](#) or sign up for a [free account](#).
- [Automation account](#) to manage machines.
- A [virtual machine](#).

Sign in to Azure

Sign in to the [Azure portal](#).

Enable the feature for deployment

1. In the [Azure portal](#), select **Virtual machines** or search for and select **Virtual machines** from the Home page.
2. Select the VM for which you want to enable Update Management. VMs can exist in any region, no matter the location of your Automation account. You
3. On the VM page, under **Operations**, select **Guest + host updates**.



4. You must have the `Microsoft.OperationalInsights/workspaces/read` permission to determine if the VM is enabled for a workspace. To learn about additional permissions that are required, see [Permissions needed to enable machines](#). To learn how to enable multiple machines at once, see [Enable Update Management from an Automation account](#).
5. On the enable Update Management page, choose the Log Analytics workspace and Automation account and click **Enable** to enable Update Management. After you've enabled Update Management, it might take about 15 minutes before you can view the update assessment from the VM.

Dashboard > CreateVm-MicrosoftWindowsServer.WindowsServer-201-20200916145019 > DC03 >

Update Management (DC03 - VM) ✖

Update Management

 Enable consistent control and compliance of this VM with Update Management.

This service is included with Azure virtual machines and Azure Arc machines. You only pay for logs stored in Log Analytics.

This service requires a Log Analytics workspace and an Automation account. You can use your existing workspace and account or let us configure the nearest workspace and account for use.

Log Analytics workspace location ⓘ
East US

Log Analytics workspace ⓘ
DefaultWorkspace38917

Automation account subscription ⓘ
Parnell Aerospace

Automation account ⓘ
Create Automation account...

Enable

Next steps

- To use Update Management for VMs, see [Manage updates and patches for your Azure VMs](#).
- To troubleshoot general Update Management errors, see [Troubleshoot Update Management issues](#).

Enable Update Management from an Automation account

8/24/2021 • 3 minutes to read • [Edit Online](#)

This article describes how you can use your Automation account to enable the [Update Management](#) feature for VMs in your environment, including machines or servers registered with [Azure Arc-enabled servers](#). To enable Azure VMs at scale, you must enable an existing Azure VM using Update Management.

NOTE

When enabling Update Management, only certain regions are supported for linking a Log Analytics workspace and an Automation account. For a list of the supported mapping pairs, see [Region mapping for Automation account and Log Analytics workspace](#).

Prerequisites

- Azure subscription. If you don't have one yet, you can [activate your MSDN subscriber benefits](#) or sign up for a [free account](#).
- [Automation account](#) to manage machines.
- An [Azure virtual machine](#), or VM or server registered with Arc-enabled servers. Non-Azure VMs or servers need to have the [Log Analytics agent](#) for Windows or Linux installed and reporting to the workspace linked to the Automation account Update Management is enabled in. We recommend installing the Log Analytics agent for Windows or Linux by first connecting your machine to [Azure Arc-enabled servers](#), and then use Azure Policy to assign the [Deploy Log Analytics agent to Linux or Windows Azure Arc machines](#) built-in policy. Alternatively, if you plan to monitor the machines with Azure Monitor for VMs, instead use the [Enable Azure Monitor for VMs](#) initiative.

Sign in to Azure

Sign in to the [Azure portal](#).

Enable Update Management

1. In your Automation account, select **Update management** under **Update management**.
2. Choose the Log Analytics workspace and Automation account and select **Enable** to enable Update Management. The setup takes up to 15 minutes to complete.

Home > Resource groups > ExampleAutoAccount > ExampleAutoAccount - Update management

ExampleAutoAccount - Update management

Automation Account

Search (Ctrl+ /) <>

CONFIGURATION MANAGEMENT

- Inventory
- Change tracking
- DSC nodes
- DSC configurations
- DSC configurations gallery (...)
- DSC node configurations

UPDATE MANAGEMENT

- Update management

PROCESS AUTOMATION

- Runbooks
- Jobs

Update Management

Enable consistent control and compliance of your VMs with Update Management.

This service is included with Azure virtual machines. You only pay for logs stored in Log Analytics.

This service requires a Log Analytics workspace and this Automation account.

Location i
East US

Log Analytics workspace subscription i
Microsoft Azure

Log Analytics workspace i
Workspace-f3aee704-2a73-4ce3-b4fb-499...

Automation account i
ExampleAutoAccount

Enable

The screenshot shows the 'Update management' configuration page in the Azure portal. On the left, there's a navigation sidebar with sections for Configuration Management (Inventory, Change tracking, DSC nodes, DSC configurations, DSC configurations gallery, DSC node configurations), Update Management (the 'Update management' item is selected and highlighted in blue), and Process Automation (Runbooks, Jobs). The main right-hand pane is titled 'Update Management' and contains descriptive text about enabling consistent control and compliance for VMs. It includes dropdown menus for Location (set to East US), Log Analytics workspace subscription (set to Microsoft Azure), and Log Analytics workspace (set to a specific workspace ID). Below these is another dropdown for the Automation account (set to ExampleAutoAccount). At the bottom is a large blue 'Enable' button.

Enable Azure VMs

1. From your Automation account select **Update management** under **Update management**.
2. Select **+ Add Azure VMs** and select one or more VMs from the list. Virtual machines that can't be enabled are grayed out and unable to be selected. Azure VMs can exist in any region no matter the location of your Automation account.
3. Select **Enable** to add the selected VMs to the computer group saved search for the feature.

Update Management

Enable consistent control and compliance of these virtual machines with Update Management.

To enable Update Management, all selected virtual machines must be in the same subscription and location. Virtual machines can be in up to 3 resource groups.

Subscription: Microsoft Azure (virtual machines: 12) | **Location:** East US (virtual machines: 12) | **Resource group:** 2 selected

Configuration:

- Log Analytics workspace: workspace-aced4c59-65db-470c-8061-2dce6fd8f3a1-eus
- Automation account: ExampleAutoAccount

Summary:

Ready to enable	Already enabled	Cannot enable
6 →	2 ✓	4 -

VM List:

NAME	UPDATE MANAGEMENT	DETAILS
linuxvm1	→ Ready to enable	
linuxvm2	● Cannot enable	VM reports to a different workspace: 'workspace-e7e3582f-a695-46cb-bc8a-b63f08f57046-eus'.
linuxvm3	→ Ready to enable	
linuxvm4	→ Ready to enable	
linuxvm5	✓ Already enabled	
linuxvm6	● Cannot enable	VM reports to a different workspace: 'workspace-e7e3582f-a695-46cb-bc8a-b63f08f57046-eus'.
winvm1	→ Ready to enable	
winvm2	● Cannot enable	VM reports to a different workspace: 'workspace-e7e3582f-a695-46cb-bc8a-b63f08f57046-eus'.
winvm3	→ Ready to enable	
winvm4	→ Ready to enable	
winvm5	✓ Already enabled	
winvm6	● Cannot enable	VM reports to a different workspace: 'workspace-e7e3582f-a695-46cb-bc8a-b63f08f57046-eus'.

Buttons: Enable | Cancel | Number of virtual machines to enable Update Management: 6

Enable non-Azure VMs

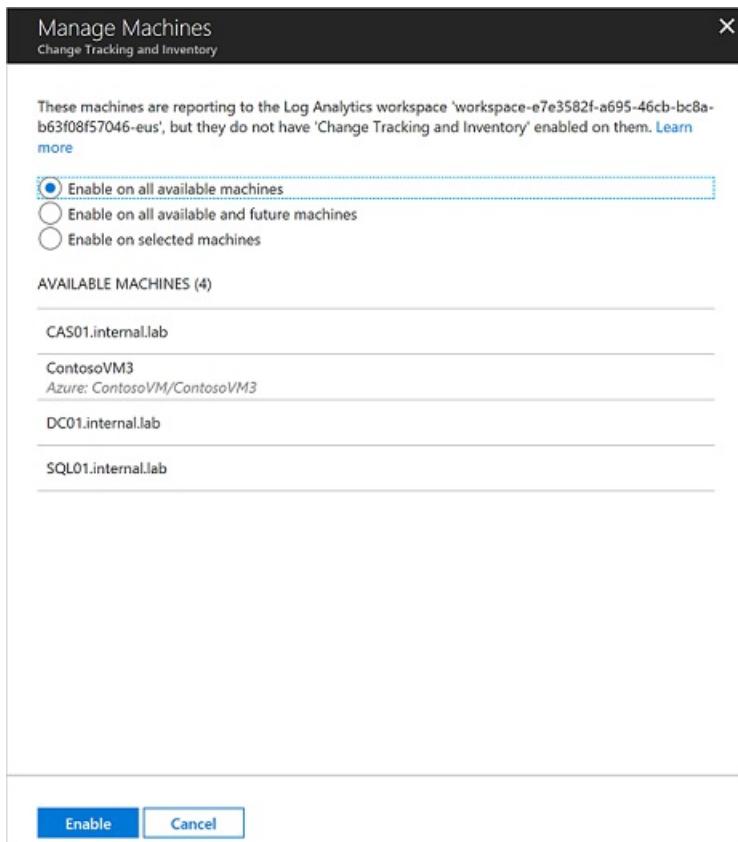
For machines or servers hosted outside of Azure, including the ones registered with Azure Arc-enabled servers, perform the following steps to enable them with Update Management.

- From your Automation account, select **Update management** under **Update management**.
- Select **Add non-Azure machine**. This action opens a new browser window with [instructions to install and configure the Log Analytics agent for Windows](#) so that the machine can begin reporting to Update Management. If you're enabling a machine that's currently managed by Operations Manager, a new agent isn't required. The workspace information is added to the agents configuration.

Enable machines in the workspace

Manually installed machines or machines already reporting to your workspace must be added to Azure Automation for Update Management to be enabled.

- From your Automation account, select **Update management** under **Update management**.
- Select **Manage machines**. The **Manage machines** button might be grayed out if you previously chose the option **Enable on all available and future machines**



- To enable Update Management for all available machines reporting to the workspace, select **Enable on all available machines** on the Manage Machines page. This action disables the control to add machines individually and adds all of the machines reporting to the workspace to the computer group saved search query `MicrosoftDefaultComputerGroup`. When selected, this action disables the **Manage Machines** option.
- To enable the feature for all available machines and future machines, select **Enable on all available and future machines**. This option deletes the saved search and scope configuration from the workspace, and permits the feature to include all Azure and non-Azure machines that currently or in the future, report to the workspace. When selected, this action disables the **Manage Machines** option permanently, as there's no scope configuration available.

NOTE

Because this option deletes the saved search and scope configuration within Log Analytics, it's important to remove any deletion locks on the Log Analytics Workspace before you select this option. If you don't, the option will fail to remove the configurations and you must remove them manually.

- If necessary, you can add the scope configuration back by re-adding the initial saved search query. For more information, see [Limit Update Management deployment scope](#).
- To enable the feature for one or more machines, select **Enable on selected machines** and select **Add** next to each machine. This task adds the selected machine names to the computer group saved search query for the feature.

Next steps

- To use Update Management for VMs, see [Manage updates and patches for your VMs](#).
- When you no longer need to manage VMs or servers with Update Management, see [remove VMs from Update Management](#).

- To troubleshoot general Update Management errors, see [Troubleshoot Update Management issues](#).

Enable Update Management from a runbook

3/5/2021 • 4 minutes to read • [Edit Online](#)

This article describes how you can use a runbook to enable the [Update Management](#) feature for VMs in your environment. To enable Azure VMs at scale, you must enable an existing VM with Update Management.

NOTE

When enabling Update Management, only certain regions are supported for linking a Log Analytics workspace and an Automation account. For a list of the supported mapping pairs, see [Region mapping for Automation account and Log Analytics workspace](#).

This method uses two runbooks:

- **Enable-MultipleSolution** - The primary runbook that prompts for configuration information, queries the specified VM and performs other validation checks, and then invokes the **Enable-AutomationSolution** runbook to configure Update Management for each VM within the specified resource group.
- **Enable-AutomationSolution** - Enables Update Management for one or more VMs specified in the target resource group. It verifies prerequisites are met, verifies the Log Analytics VM extension is installed and installs if not found, and adds the VMs to the scope configuration in the specified Log Analytics workspace linked to the Automation account.

Prerequisites

- Azure subscription. If you don't have one yet, you can [activate your MSDN subscriber benefits](#) or sign up for a [free account](#).
- [Automation account](#) to manage machines.
- [Log Analytics workspace](#)
- A [virtual machine](#).
- Two Automation assets, which are used by the **Enable-AutomationSolution** runbook. This runbook, if it doesn't already exist in your Automation account, is automatically imported by the **Enable-MultipleSolution** runbook during its first run.
 - *LASolutionSubscriptionId*: Subscription ID of where the Log Analytics workspace is located.
 - *LASolutionWorkspaceId*: Workspace ID of the Log Analytics workspace linked to your Automation account.

These variables are used to configure the workspace of the onboarded VM, and you need to manually create them. If these are not specified, the script first searches for any VM onboarded to Update Management in its subscription, followed by the subscription the Automation account is in, followed by all other subscriptions your user account has access to. If not properly configured, this may result in your machines getting onboarded to some random Log Analytics workspace.

Sign in to Azure

Sign in to the [Azure portal](#).

Enable Update Management

1. In the Azure portal, navigate to **Automation Accounts**. On the **Automation Accounts** page, select your account from the list.
2. In your Automation account, select **Update Management** under **Update Management**.
3. Select the Log Analytics workspace, then click **Enable**. While Update Management is being enabled, a banner is shown.

The screenshot shows the 'Update management' configuration page for the 'Prod1-AA-Sec' automation account. The left sidebar lists 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', 'Diagnose and solve problems', 'Configuration Management' (with 'Inventory', 'Change tracking', and 'State configuration (DSC)'), 'Update management' (which is selected), 'Process Automation' (with 'Runbooks', 'Jobs', and 'Runbooks gallery'), and 'Shared Resources'. The main panel title is 'Update Management' with a subtitle 'Enable consistent control and compliance of your VMs with Update Management.' It states that the service is included with Azure virtual machines and Azure Arc machines. It requires a Log Analytics workspace and the Automation account. A dropdown for 'Log Analytics workspace location' is set to 'East US'. Another dropdown for 'Log Analytics workspace subscription' is set to 'Contoso'. A third dropdown for 'Log Analytics workspace' is set to 'Prod1-LA-Sec'. An 'Automation account' dropdown is set to 'Prod1-AA-Sec'. At the bottom is a large blue 'Enable' button.

Install and update modules

It's required to update to the latest Azure modules and import the [AzureRM.OperationalInsights](#) module to successfully enable Update Management for your VMs with the runbook.

1. In your Automation account, select **Modules** under **Shared Resources**.
2. Select **Update Azure Modules** to update the Azure modules to the latest version.
3. Click **Yes** to update all existing Azure modules to the latest version.

The screenshot shows the 'Modules' section for the 'MAIC-AA-Pri' automation account. The left sidebar lists 'Jobs', 'Runbooks gallery', 'Hybrid worker groups', 'Watcher tasks', 'Shared Resources' (with 'Schedules' and 'Modules'), and 'Modules' (which is selected). The main panel has a search bar 'Search modules...'. Below it is a table with columns 'Name', 'Last modified', and 'Status'. The table lists several Azure modules: AuditPolicyDsc (9/16/2020, 4:15 AM, Available), Az.Accounts (6/19/2020, 4:07 PM, Available), Az.Automation (3/17/2020, 9:40 AM, Available), Az.Compute (6/19/2020, 4:09 PM, Available), Azure (11/7/2019, 4:13 PM, Available), and Azure.Storage (1/31/2020, 3:14 PM, Available). At the top of the main panel are buttons: '+ Add a module', 'Update Azure modules', 'Learn about module updates', 'Browse gallery' (which is highlighted in blue), and 'Refresh'.

Name	Last modified	Status
AuditPolicyDsc	9/16/2020, 4:15 AM	Available
Az.Accounts	6/19/2020, 4:07 PM	Available
Az.Automation	3/17/2020, 9:40 AM	Available
Az.Compute	6/19/2020, 4:09 PM	Available
Azure	11/7/2019, 4:13 PM	Available
Azure.Storage	1/31/2020, 3:14 PM	Available

4. Return to **Modules** under **Shared Resources**.
5. Select **Browse gallery** to open the module gallery.

6. Search for `AzureRM.OperationalInsights` and import this module into your Automation account.

The screenshot shows the Azure PowerShell Gallery interface. On the left, there's a search bar with 'AzureRM.Operational' and a sidebar with a 'Browse Gallery' button. The main area displays the 'AzureRM.OperationalInsights' module details:

- Name:** AzureRM.OperationalInsights
- Description:** PowerShell Module
- Import:** Import
- Summary:** Microsoft Azure PowerShell - OperationalInsights service cmdlets for Azure Resource Manager
- Created by:** azure-sdk
- Tags:** Azure ResourceManager ARM OperationalInsights PSModule
- Dependencies:** AzureRM.Profile (≥ 5.5.1)
- View Source Project**
- Learn more:** View in PowerShell Gallery, Documentation, Licensing information
- Content:** A table showing cmdlets:

Type	Name
Cmdlet	New-AzureRmOperationalInsightsAzureActivityLogDataSource
Cmdlet	New-AzureRmOperationalInsightsCustomLogDataSource
Cmdlet	Disable-AzureRmOperationalInsightsLinuxCustomLogCollection
- Version:** 5.0.6
- Downloads:** 11,639,598
- Last updated:** 8/29/2018

Select Azure VM to manage

With Update Management enabled, you can add an Azure VM to receive updates.

1. From your Automation account, select **Update management** under the section **Update management**.
2. Select **Add Azure VMs** to add your VM.
3. Choose the VM from the list and click **Enable** to configure the VM for management.

The screenshot shows the 'Enable Update Management' blade in the Azure portal. It has the following sections:

- Update Management:** Enable consistent control and compliance of these virtual machines with Update Management.
- Subscription:** Contoso (virtual machines: 13)
- Location:** West US (virtual machines: 1)
- Resource groups:** prodwus
- Configuration (used when enabling new VMs):** AUTO: Auto-configure Log Analytics workspace and Automation account based on VMs subscription and location. CUSTOM: Choose existing Log Analytics workspace and Automation account.
- Log Analytics workspace:** prodwus-pri-la
- Automation account:** ProdWUS-Pri-AA
- Summary:** Ready to enable (1), Already enabled (0), Cannot enable (0).
- Table:** Shows one VM: prodwus-dc01, status: Ready to enable.
- Buttons:** Enable (blue), Cancel.

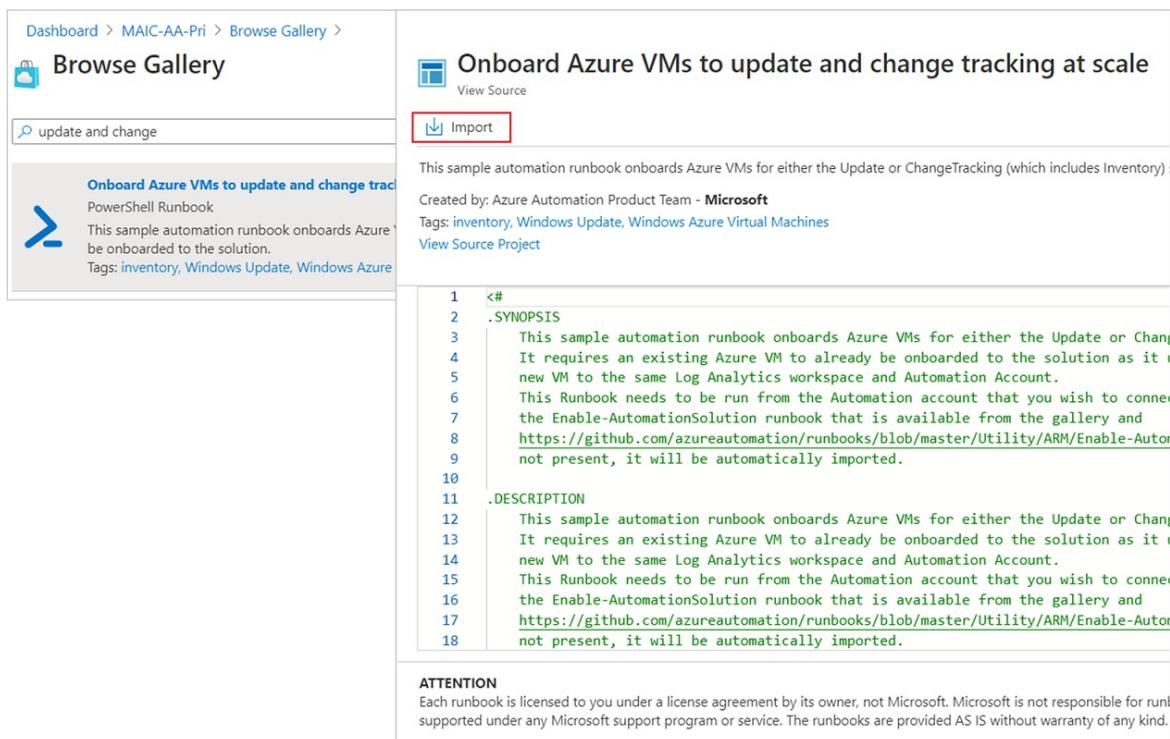
NOTE

If you try to enable another feature before setup of Update Management has completed, you receive this message:

Installation of another solution is in progress on this or a different virtual machine. When that installation completes the Enable button is enabled, and you can request installation of the solution on this virtual machine.

Import a runbook to enable Update Management

1. In your Automation account, select **Runbooks** under Process Automation.
2. Select **Browse gallery**.
3. Search for **update and change tracking**.
4. Select the runbook and click **Import** on the **View Source** page.
5. Click **OK** to import the runbook into the Automation account.



The screenshot shows two side-by-side views of the Azure Automation interface. On the left is the 'Browse Gallery' page, where a search for 'update and change' has returned the 'Onboard Azure VMs to update and change tracking' runbook. This runbook is described as a PowerShell runbook that onboards Azure VMs for either Update or Change Tracking. It has been created by the Azure Automation Product Team - Microsoft and includes tags for inventory, Windows Update, and Windows Azure. On the right is the 'View Source' page for the same runbook. The source code is displayed in a monospaced font, starting with a synopsis that explains the runbook's purpose of onboarding Azure VMs for Update or Change Tracking. It requires an existing Azure VM to be onboarded to the solution as it creates a new VM to the same Log Analytics workspace and Automation Account. The runbook needs to be run from the Automation account that wishes to connect to the target VMs. The source code also includes a link to the GitHub repository for the runbook. At the bottom of the source page is an 'ATTENTION' section stating that each runbook is licensed under a license agreement by its owner, not Microsoft, and that Microsoft is not responsible for runbooks supported under any Microsoft support program or service. The runbooks are provided AS IS without warranty of any kind.

6. On the **Runbook** page, select the **Enable-MultipleSolution** runbook and then click **Edit**. On the textual editor, select **Publish**.
7. When prompted to confirm, click **Yes** to publish the runbook.

Start the runbook

You must have enabled Update Management for an Azure VM to start this runbook. It requires an existing VM and resource group with the feature enabled in order to configure one or more VMs in the target resource group.

1. Open the **Enable-MultipleSolution** runbook.

The screenshot shows the Azure portal interface for a runbook named 'Enable-MultipleSolution'. The top navigation bar includes 'Start', 'View', 'Edit', 'Link to schedule', 'Add webhook', 'Delete', and 'Export' buttons. The main content area is titled 'Essentials' and displays resource group (ProdWUS), account (ProdWUS-Pri-AA), location (West US 2), and subscription (Contoso). It also shows a section for 'Tags' with a link to add tags. Below this is a 'Recent Jobs' section with columns for Status, Created, and Last updated, which currently shows 'No jobs found.' On the left sidebar, there are sections for Overview, Activity log, Tags, Diagnose and solve problems, Resources (Jobs, Schedules, Webhooks), Runbook settings (Properties, Description, Logging and tracing), and a search bar at the top.

2. Click the start button and enter parameter values in the following fields:

- **VMNAME** - The name of an existing VM to add to Update Management. Leave this field blank to add all VMs in the resource group.
- **VMRESOURCEGROUP** - The name of the resource group for the VMs to enable.
- **SUBSCRIPTIONID** - The subscription ID of the new VM to enable. Leave this field blank to use the subscription of the workspace. When you use a different subscription ID, add the Run As account for your Automation account as a contributor for the subscription.
- **ALREADYONBOARDEDVM** - The name of the VM that is already manually enabled for updates.
- **ALREADYONBOARDEDVMRESOURCEGROUP** - The name of the resource group to which the VM belongs.
- **SOLUTIONTYPE** - Enter **Updates**.

Start Runbook ✎ X

Enable-MultipleSolution

Parameters

VMNAME ⓘ
No value
Optional, String

VMRESOURCEGROUP * ⓘ
Finance ✓
Mandatory, String

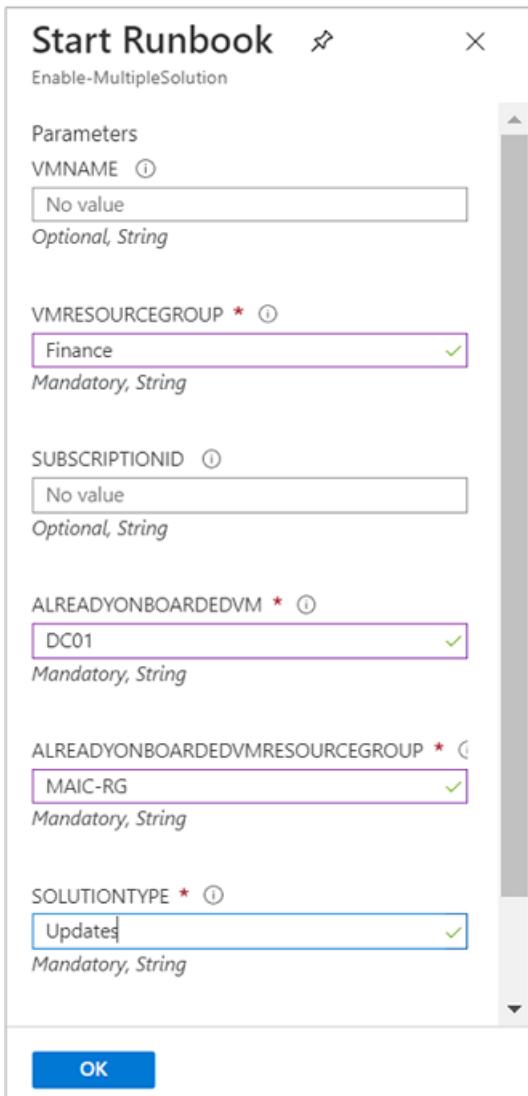
SUBSCRIPTIONID ⓘ
No value
Optional, String

ALREADYONBOARDEDVM * ⓘ
DC01 ✓
Mandatory, String

ALREADYONBOARDEDVMRESOURCEGROUP * ⓘ
MAIC-RG ✓
Mandatory, String

SOLUTIONTYPE * ⓘ
Updates ✓
Mandatory, String

OK



3. Select **OK** to start the runbook job.
4. Monitor progress of the runbook job and any errors from the **Jobs** page.

Next steps

- To use Update Management for VMs, see [Manage updates and patches for your VMs](#).
- To troubleshoot general Update Management errors, see [Troubleshoot Update Management issues](#).

Manage updates and patches for your VMs

3/5/2021 • 2 minutes to read • [Edit Online](#)

Software updates in Azure Automation Update Management provides a set of tools and resources that can help manage the complex task of tracking and applying software updates to machines in Azure and hybrid cloud. An effective software update management process is necessary to maintain operational efficiency, overcome security issues, and reduce the risks of increased cyber security threats. However, because of the changing nature of technology and the continual appearance of new security threats, effective software update management requires consistent and continual attention.

NOTE

Update Management supports the deployment of first-party updates and the pre-downloading of them. This support requires changes on the systems being updated. See [Configure Windows Update settings for Azure Automation Update Management](#) to learn how to configure these settings on your systems.

Before attempting to manage updates for your VMs, ensure that you've enabled Update Management on them using one of these methods:

- [Enable Update Management from an Automation account](#)
- [Enable Update Management by browsing the Azure portal](#)
- [Enable Update Management from a runbook](#)
- [Enable Update Management from an Azure VM](#)

Limit the scope for the deployment

Update Management uses a scope configuration within the workspace to target the computers to receive updates. For more information, see [Limit Update Management deployment scope](#).

Compliance assessment

Before you deploy software updates to your machines, review the update compliance assessment results for enabled machines. For each software update, its compliance state is recorded and then after the evaluation is complete, it is collected and forwarded in bulk to Azure Monitor logs.

On a Windows machine, the compliance scan is run every 12 hours by default, and is initiated within 15 minutes of the Log Analytics agent for Windows is restarted. The assessment data is then forwarded to the workspace and refreshes the **Updates** table. Before and after update installation, an update compliance scan is performed to identify missing updates, but the results are not used to update the assessment data in the table.

It is important to review our recommendations on how to [configure the Windows Update client](#) with Update Management to avoid any issues that prevents it from being managed correctly.

For a Linux machine, the compliance scan is performed every hour by default. If the Log Analytics agent for Linux is restarted, a compliance scan is initiated within 15 minutes.

The compliance results are presented in Update Management for each machine assessed. It can take up to 30 minutes for the dashboard to display updated data from a new machine enabled for management.

Review [monitor software updates](#) to learn how to view compliance results.

Deploy updates

After reviewing the compliance results, the software update deployment phase is the process of deploying software updates. To install updates, schedule a deployment that aligns with your release schedule and service window. You can choose which update types to include in the deployment. For example, you can include critical or security updates and exclude update rollups.

Review [deploy software updates](#) to learn how to schedule an update deployment.

Review update deployments

After the deployment is complete, review the process to determine the success of the update deployment by machine or target group. See [review deployment status](#) to learn how you can monitor the deployment status.

Next steps

- To learn how to create alerts to notify you about update deployment results, see [create alerts for Update Management](#).
- You can [query Azure Monitor logs](#) to analyze update assessments, deployments, and other related management tasks. It includes pre-defined queries to help you get started.

View update assessments in Update Management

8/24/2021 • 3 minutes to read • [Edit Online](#)

In Update Management, you can view information about your machines, missing updates, update deployments, and scheduled update deployments. You can view the assessment information scoped to the selected Azure virtual machine, from the selected Arc-enabled server, or from the Automation account across all configured machines and servers.

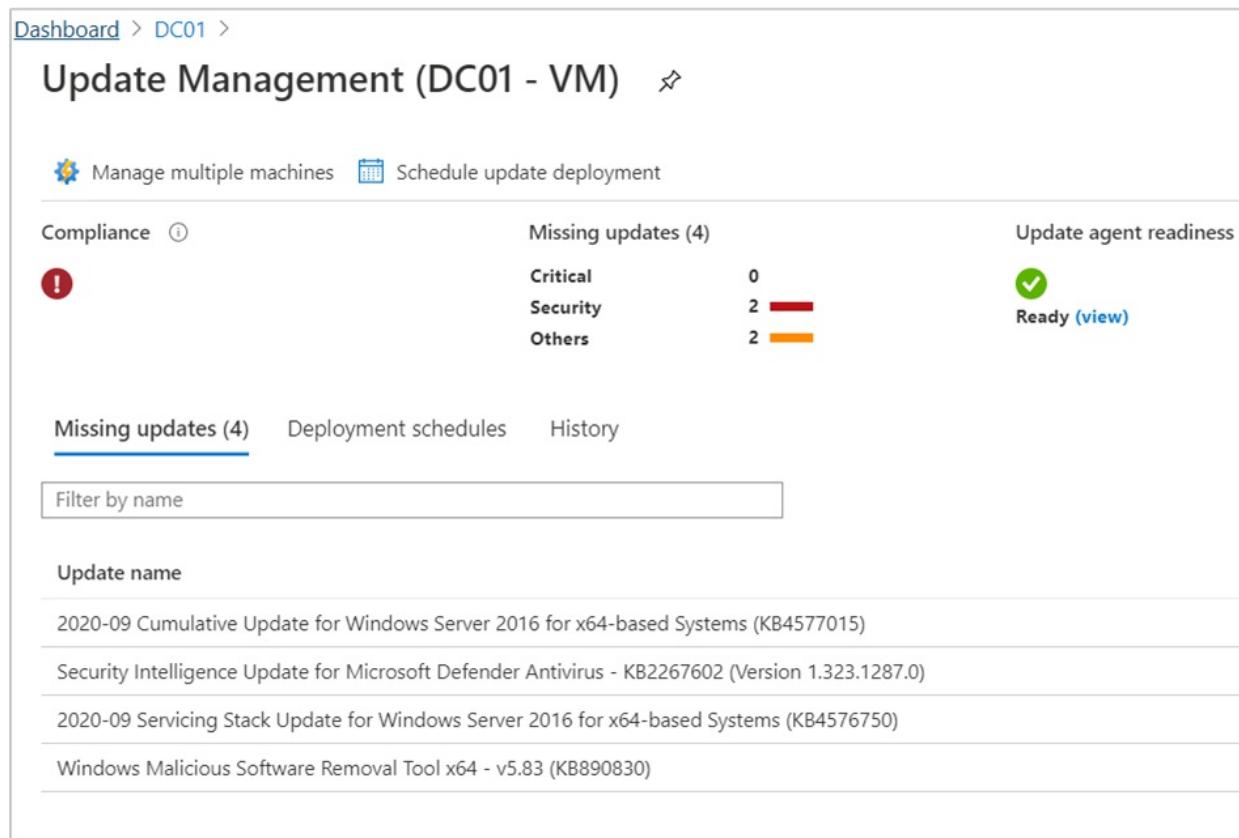
Sign in to the Azure portal

[Sign in to the Azure portal](#)

View update assessment

To view update assessment from an Azure VM, navigate to **Virtual Machines** and select your virtual machine from the list. From the left menu, select **Guest + host updates**, and then select **Go to Update Management** on the **Guest + host updates** page.

In Update Management, you can view information about your machine, missing updates, update deployments, and scheduled update deployments.



The screenshot shows the Azure Update Management interface for the VM 'DC01'. At the top, there are links for 'Dashboard > DC01 > Update Management (DC01 - VM)'. Below this, there are two buttons: 'Manage multiple machines' and 'Schedule update deployment'. A 'Compliance' section shows a red exclamation mark icon, indicating a critical issue. The 'Missing updates (4)' section lists three categories: Critical (0), Security (2), and Others (2). To the right, 'Update agent readiness' is shown as 'Ready (view)' with a green checkmark icon. Below these sections, there are tabs for 'Missing updates (4)', 'Deployment schedules', and 'History', with 'Missing updates (4)' being the active tab. A 'Filter by name' input field is present. Under the 'Update name' heading, a list of four updates is displayed:

- 2020-09 Cumulative Update for Windows Server 2016 for x64-based Systems (KB4577015)
- Security Intelligence Update for Microsoft Defender Antivirus - KB2267602 (Version 1.323.1287.0)
- 2020-09 Servicing Stack Update for Windows Server 2016 for x64-based Systems (KB4576750)
- Windows Malicious Software Removal Tool x64 - v5.83 (KB890830)

To view update assessment from an Arc-enabled server, navigate to **Servers - Azure Arc** and select your server from the list. From the left menu, select **Guest and host updates**. On the **Guest + host updates** page, select **Go to Update Management**.

In Update Management, you can view information about your Arc enabled machine, missing updates, update deployments, and scheduled update deployments.

The screenshot shows the Azure portal interface for the server FNPSVR01 under the 'Server - Azure Arc' category. The main header is 'FNPSVR01 | Update management'. On the left, there's a navigation menu with links like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings (Security, Extensions, Properties, Locks), and a search bar. The central area shows a green checkmark icon indicating 'Compliance' status. Below it, 'Missing updates (0)' is shown with breakdowns for Critical (0), Security (0), and Others (0). There are tabs for Missing updates (0), Deployment schedules, and History, with 'Missing updates (0)' being the active tab. A 'Filter by name' input field is present. The bottom section shows a summary of non-compliant machines (4 out of 6) and machines needing attention (6), with a bar chart for critical and security updates. Below this is a table listing individual machine details: SQL01.internal.lab, CAS01.internal.lab, DC01.internal.lab, WinVM6, LinuxVM2, and LinuxVM6. Each row includes columns for Machine Name, Compliance (with a link to the last assessment), and Platform (Azure).

To view update assessment across all machines, including Arc-enabled servers from your Automation account, navigate to **Automation accounts** and select your Automation account with Update Management enabled from the list. In your Automation account, select **Update management** from the left menu.

The updates for your environment are listed on the **Update management** page. If any updates are identified as missing, a list of them is shown on the **Missing updates** tab.

This screenshot shows the 'Missing updates' tab from the previous page. At the top, there are buttons for 'Schedule update deployment', 'Add Azure VMs', and 'Add non-Azure machine'. Below that, two sections are displayed: 'Non-compliant machines' (4 out of 6) and 'Machines need attention (6)'. The 'Machines need attention' section includes a bar chart for 'Critical and security' (4) and 'Other' (2) updates. The main table lists 6 machines with their names, compliance status (all are Non-compliant), and platform (all are Azure). The 'COMPLIANCE' column contains a link to the last assessment time.

MACHINE NAME	COMPLIANCE	PLATFORM
SQL01.internal.lab	! Non-compliant as of 6/14/2018, 10:29 AM	Azure
CAS01.internal.lab	! Non-compliant as of 6/14/2018, 10:48 AM	Azure
DC01.internal.lab	! Non-compliant as of 6/14/2018, 5:23 AM	Azure
WinVM6	! Non-compliant as of 6/14/2018, 11:10 AM	Azure
LinuxVM2	✓ Compliant as of 6/14/2018, 11:09 AM	Azure
LinuxVM6	✓ Compliant as of 6/14/2018, 11:09 AM	Azure

Under the **COMPLIANCE** column, you can see the last time the machine was assessed. Under the **UPDATE AGENT READINESS** column, you can see the health of the update agent. If there's an issue, select the link to go to troubleshooting documentation that can help you correct the problem.

Under **Information link**, select the link for an update to open the support article that gives you important information about the update.

Home > Automation Accounts >

MAIC-AA-Pri | Update management

Automation Account

Search (Ctrl+ /) Schedule update deployment Add Azure VMs Add non-

Overview Activity log Access control (IAM) Tags Diagnose and solve problems

Configuration Management

- Inventory
- Change tracking
- State configuration (DSC)

Update management

- Update management

Process Automation

- Runbooks

Non-compliant machines 1 out of 4 Machines need attention (2)

Critical and security	1
Other	1
Not assessed	0

Machines (4) Missing updates (9) Deployment schedules

Filter by name

Update name

- AzureConnectedMachineAgent - July 2020 Release
- 2020-07 Cumulative Update for Windows Server 2019 (1809) for x64-based Systems (KB4496411)
- 2020-06 Security Update for Adobe Flash Player for Windows Server 2019 (KB4496410)
- 2020-07 Cumulative Update Preview for .NET Framework 3.5, 4.7.2 and 4.8 (KB4496409)
- Security Intelligence Update for Microsoft Defender Antivirus - KB2267603
- 2020-01 Update for Windows Server 2019 for x64-based Systems (KB4496408)
- 2020-03 Cumulative Update for Windows Server 2019 (1809) for x64-based Systems (KB4496407)

NOTE

Information that is displayed about the Windows Defender definition update status is based on the last data that was summarized from the Log Analytics workspace and might not be current. Review [Windows Defender update always show as missing](#) to learn more about this behavior.

Click anywhere else on the update to open the Log Search pane. The query for the log search is predefined for that specific update. You can modify this query or create your own query to view detailed information.

```

Home > Automation Accounts > MAIC-AA-Pri | Update management >
Logs MAIC-LA
New Query 1* +
MAIC-LA Select scope Run Time range : S
Update
| where TimeGenerated>ago(14h) and OSType!="Linux" and (OSType=="Windows" and TimeGenerated>ago(12h))
| summarize arg_max(TimeGenerated, Solutions) by SourceComputerId
| where Solutions has "updates"
| distinct SourceComputerId)
| summarize hint.strategy=partitioned arg_max(TimeGenerated, Solutions)
| where UpdateState=~"Needed" and Approved!=false and Title!=""
| render table

```

Results Chart Columns ▾ Display time (UTC+00:00)

Completed

TimeGenerated [UTC]	Computer	SourceComputerId
7/30/2020, 3:02:23.533 PM	SVR03.Corp.Swanson.Local	1a64454c-6689-...

View missing updates

Select **Missing updates** to view the list of updates that are missing from your machines. Each update is listed and can be selected. Information about the number of machines that require the update, operating system details, and a link for more information are all shown. The Log Search pane also shows more details about the updates.

Schedule update deployment + Add Azure VMs ⚙ Add non-Azure machine ⚙ Manage machines

Non-compliant machines 1 out of 4

Machines need attention (2)	Missing updates (9)	Failed update deployments (0)
Critical and security 1	Critical 1	0
Other 1	Security 2	out of 1 in the past six months
Not assessed 0	Others 6	

Machines (4) Missing updates (9) Deployment schedules History

Filter by name Classifications: All

Update name	Classification	Machines missing upda...	Operating system
AzureConnectedMachineAgent - July 2020 Release	✖ Critical updates	1	Windows
2020-07 Cumulative Update for Windows Server 2019 (1809) for x64-bas...	⚠ Security updates	1	Windows
2020-06 Security Update for Adobe Flash Player for Windows Server 2019...	⚠ Security updates	1	Windows
2020-07 Cumulative Update Preview for .NET Framework 3.5, 4.7.2 and 4....	⚠ Updates	1	Windows
Security Intelligence Update for Microsoft Defender Antivirus - KB226760...	⚠ Definition updates	2	Windows
2020-01 Update for Windows Server 2019 for x64-based Systems (KB449...	⚠ Updates	1	Windows
2020-03 Cumulative Update for Windows Server 2019 (1809) for x64-bas...	⚠ Updates	1	Windows

Work with update classifications

The following tables list the supported update classifications in Update Management, with a definition for each classification.

Windows

CLASSIFICATION	DESCRIPTION
Critical updates	Updates for specific problems that address critical, non-security-related bugs.
Security updates	Updates for product-specific, security-related issues.
Update rollups	Sets of hotfixes that are packaged together for easy deployment.
Feature packs	New product features that are distributed outside a product release.
Service packs	Sets of hotfixes that are applied to an application.
Definition updates	Updates to virus or other definition files.
Tools	Utilities or features that help complete one or more tasks.
Updates	Updates to applications or files that are installed currently.

Linux

CLASSIFICATION	DESCRIPTION
Critical and security updates	Updates for a specific problem or a product-specific, security-related issue.
Other updates	All other updates that aren't critical in nature or that aren't security updates.

For Linux, Update Management can distinguish between critical updates and security updates in the cloud while displaying assessment data. (This granularity is possible because of data enrichment in the cloud.) For patching, Update Management relies on classification data available on the machine. Unlike other distributions, CentOS doesn't have this information available in the RTM versions of the product. If you have CentOS machines configured to return security data for the following command, Update Management can patch based on classifications:

```
sudo yum -q --security check-update
```

There's currently no supported method to enable native classification-data availability on CentOS. At this time, only best-effort support is provided to customers who have enabled this functionality on their own.

To classify updates on Red Hat Enterprise version 6, you need to install the yum-security plugin. On Red Hat Enterprise Linux 7, the plugin is already a part of yum itself, there is no need to install anything. For further information, see the following Red Hat [knowledge article](#).

Next steps

The next phase in the process is to [deploy updates](#) to non-compliant machines and review deployment results.

How to deploy updates and review results

8/24/2021 • 9 minutes to read • [Edit Online](#)

This article describes how to schedule an update deployment and review the process after the deployment is complete. You can configure an update deployment from a selected Azure virtual machine, from the selected Arc-enabled server, or from the Automation account across all configured machines and servers.

Under each scenario, the deployment you create targets that selected machine or server, or in the case of creating a deployment from your Automation account, you can target one or more machines. When you schedule an update deployment from an Azure VM or Arc-enabled server, the steps are the same as deploying from your Automation account, with the following exceptions:

- The operating system is automatically pre-selected based on the OS of the machine
- The target machine to update is set to target itself automatically
- When configuring the schedule, you can specify **Update now**, occurs once, or uses a recurring schedule.

IMPORTANT

By creating an update deployment, you accept the terms of the Software License Terms (EULA) provided by the company offering updates for their operating system.

Sign in to the Azure portal

Sign in to the [Azure portal](#)

Schedule an update deployment

Scheduling an update deployment creates a [schedule](#) resource linked to the **Patch-MicrosoftOMSComputers** runbook that handles the update deployment on the target machine or machines. You must schedule a deployment that follows your release schedule and service window to install updates. You can choose the update types to include in the deployment. For example, you can include critical or security updates, and exclude update rollups.

NOTE

If you delete the schedule resource from the Azure portal or using PowerShell after creating the deployment, the deletion breaks the scheduled update deployment and presents an error when you attempt to reconfigure the schedule resource from the portal. You can only delete the schedule resource by deleting the corresponding deployment schedule.

To schedule a new update deployment, perform the following steps. Depending on the resource selected (that is, Automation account, Arc-enabled server, Azure VM), the steps below apply to all with minor differences while configuring the deployment schedule.

1. In the portal, to schedule a deployment for:

- One or more machines, navigate to **Automation accounts** and select your Automation account with Update Management enabled from the list.
- For an Azure VM, navigate to **Virtual machines** and select your VM from the list.
- For an Arc-enabled server, navigate to **Servers - Azure Arc** and select your server from the list.

2. Depending on the resource you selected, to navigate to Update Management:

- If you selected your Automation account, go to **Update management** under **Update management**, and then select **Schedule update deployment**.
- If you selected an Azure VM, go to **Guest + host updates**, and then select **Go to Update Management**.
- If you selected an Arc-enabled server, go to **Update Management**, and then select **Schedule update deployment**.

3. Under **New update deployment**, in the **Name** field enter a unique name for your deployment.

4. Select the operating system to target for the update deployment.

NOTE

This option is not available if you selected an Azure VM or Arc-enabled server. The operating system is automatically identified.

5. In the **Groups to update** region, define a query that combines subscription, resource groups, locations, and tags to build a dynamic group of Azure VMs to include in your deployment. To learn more, see [Use dynamic groups with Update Management](#).

NOTE

This option is not available if you selected an Azure VM or Arc-enabled server. The machine is automatically targeted for the scheduled deployment.

IMPORTANT

When building a dynamic group of Azure VMs, Update Management only supports a maximum of 500 queries that combines subscriptions or resource groups in the scope of the group.

6. In the **Machines to update** region, select a saved search, an imported group, or pick **Machines** from the dropdown menu and select individual machines. With this option, you can see the readiness of the Log Analytics agent for each machine. To learn about the different methods of creating computer groups in Azure Monitor logs, see [Computer groups in Azure Monitor logs](#). You can include up to a maximum of 1000 machines in a scheduled update deployment.

NOTE

This option is not available if you selected an Azure VM or Arc-enabled server. The machine is automatically targeted for the scheduled deployment.

7. Use the **Update classifications** region to specify [update classifications](#) for products. For each product, deselect all supported update classifications but the ones to include in your update deployment.

New update deployment

Update Deployment

Groups to update >
1 group selected

Machines to update >
Click to Configure

Update classifications

3 selected ^

- Select all
- Critical updates
- Security updates
- Update rollups
- Feature packs
- Service packs
- Definition updates
- Tools
- Updates

Create

If your deployment is meant to apply only a select set of updates, it is necessary to deselect all the pre-selected update classifications when configuring the **Include/exclude updates** option as described in the next step. This ensures only the updates you have specified to *include* in this deployment are installed on the target machines.

NOTE

Deploying updates by update classification doesn't work on RTM versions of CentOS. To properly deploy updates for CentOS, select all classifications to make sure updates are applied. There's currently no supported method to enable native classification-data availability on CentOS. See the following for more information about [Update classifications](#).

NOTE

Deploying updates by update classification may not work correctly for Linux distros supported by Update Management. This is a result of an issue identified with the naming schema of the OVAL file and this prevents Update Management from properly matching classifications based on filtering rules. Because of the different logic used in security update assessments, results may differ from the security updates applied during deployment. If you have classification set as **Critical** and **Security**, the update deployment will work as expected. Only the *classification of updates* during an assessment is affected.

Update Management for Windows Server machines is unaffected; update classification and deployments are unchanged.

8. Use the **Include/exclude updates** region to add or exclude selected updates from the deployment. On the **Include/Exclude** page, you enter KB article ID numbers to include or exclude for Windows updates. For supported Linux distros, you specify the package name.

Include/exclude updates ...

[Include](#) [Exclude](#)

Provide a list of KBs (without the 'KB' prefix) that should be specifically added during the update deployment.



KBIDs

5001648

5001633

Enter KB id, e.g. 168934

[OK](#)

[Cancel](#)

IMPORTANT

Remember that exclusions override inclusions. For instance, if you define an exclusion rule of *, Update Management excludes all patches or packages from the installation. Excluded patches still show as missing from the machines. For Linux machines, if you include a package that has a dependent package that has been excluded, Update Management doesn't install the main package.

NOTE

You can't specify updates that have been superseded to include in the update deployment.

Here are some example scenarios to help you understand how to use inclusion/exclusion and update classification simultaneously in update deployments:

- If you only want to install a specific list of updates, you should not select any **Update classifications** and provide a list of updates to be applied using **Include** option.
- If you want to install only security and critical updates, along with one or more optional driver updates, you should select **Security** and **Critical** under **Update classifications**. Then for the **Include** option, specify the driver updates.
- If you want to install only security and critical updates, but skip one or more updates for python to avoid breaking your legacy application, you should select **Security** and **Critical** under **Update classifications**. Then for the **Exclude** option add the python packages to skip.

9. Select **Schedule settings**. The default start time is 30 minutes after the current time. You can set the start time to any time from 10 minutes in the future.

NOTE

This option is different if you selected an Arc-enabled server. You can select **Update now** or a start time 20 minutes into the future.

10. Use the **Recurrence** to specify if the deployment occurs once or uses a recurring schedule, then select **OK**.
 11. In the **Pre-scripts + Post-scripts** region, select the scripts to run before and after your deployment. To learn more, see [Manage pre-scripts and post-scripts](#).
 12. Use the **Maintenance window (minutes)** field to specify the amount of time allowed for updates to install. Consider the following details when specifying a maintenance window:
 - Maintenance windows control how many updates are installed.
 - Update Management doesn't stop installing new updates if the end of a maintenance window is approaching.
 - Update Management doesn't terminate in-progress updates if the maintenance window is exceeded. Any remaining updates to be installed are not attempted. If this is consistently happening, you should reevaluate the duration of your maintenance window.
 - If the maintenance window is exceeded on Windows, it's often because a service pack update is taking a long time to install.
- NOTE**

To avoid updates being applied outside of a maintenance window on Ubuntu, reconfigure the `Unattended-Upgrade` package to disable automatic updates. For information about how to configure the package, see the [Automatic updates topic in the Ubuntu Server Guide](#).
13. Use the **Reboot options** field to specify the way to handle reboots during deployment. The following options are available:
 - Reboot if necessary (default)
 - Always reboot
 - Never reboot
 - Only reboot; this option doesn't install updates
- NOTE**

The registry keys listed under [Registry keys used to manage restart](#) can cause a reboot event if **Reboot options** is set to **Never reboot**.
14. When you're finished configuring the deployment schedule, select **Create**.

New update deployment

Update Deployment

Name * ⓘ

September Windows Updates ✓

Operating system

Windows Linux

* Items to update

Add groups and/or machines to update

Groups to update >

Click to Configure

Machines to update >

1 items selected

Update classifications

8 selected ▾

Include/exclude updates >

Click to Configure

*Schedule settings >

One time

Create

NOTE

When you're finished configuring the deployment schedule for a selected Arc-enabled server, select **Review + create**.

15. You're returned to the status dashboard. Select **Deployment schedules** to show the deployment schedule that you've created. A maximum of 500 schedules are listed. If you have more than 500 schedules and you want to review the complete list, see the [Software Update Configurations - List](#) REST API method. Specify API version 2019-06-01 or higher.

Schedule an update deployment programmatically

To learn how to create an update deployment with the REST API, see [Software Update Configurations - Create](#).

You can use a sample runbook to create a weekly update deployment. To learn more about this runbook, see [Create a weekly update deployment for one or more VMs in a resource group](#).

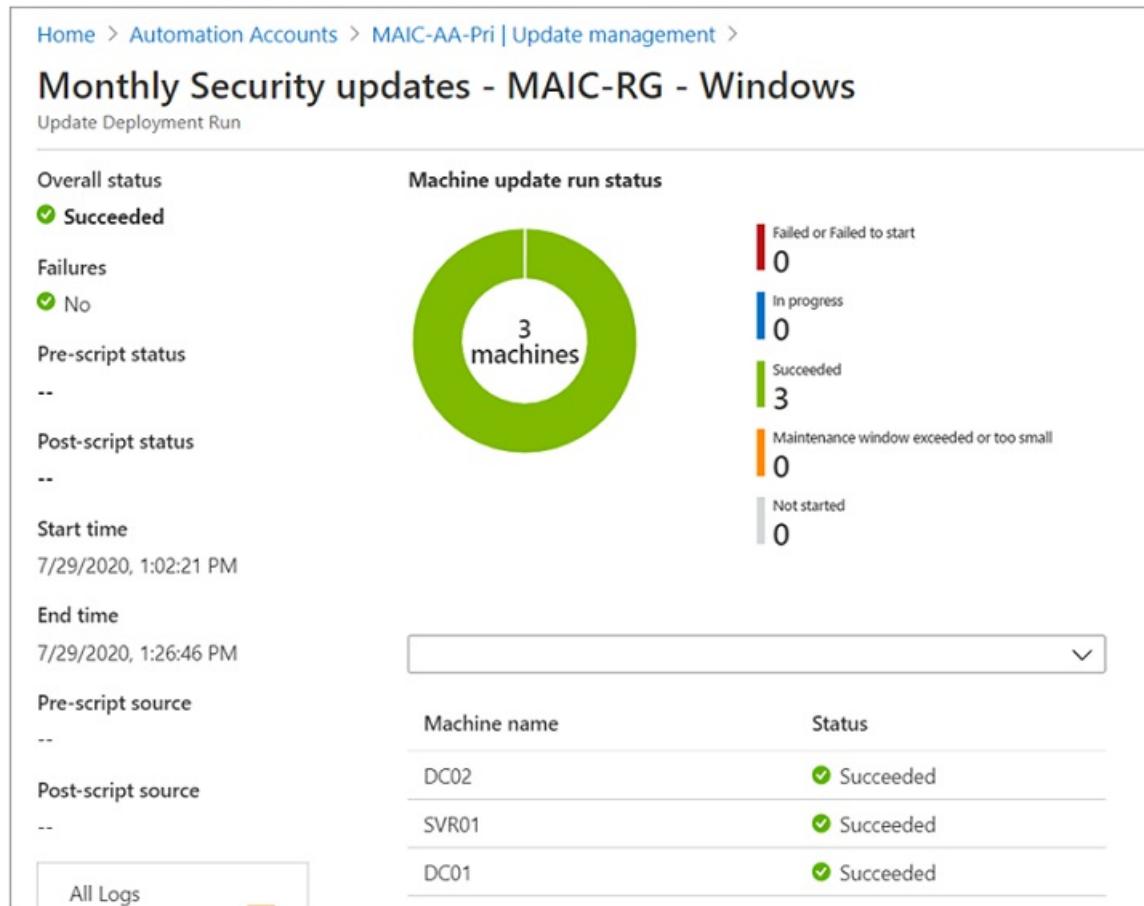
Check deployment status

After your scheduled deployment starts, you can see its status on the **History** tab under **Update management**. The status is **In progress** when the deployment is currently running. When the deployment ends successfully, the status changes to **Succeeded**. If there are failures with one or more updates in the

deployment, the status is Failed.

View results of a completed update deployment

When the deployment is finished, you can select it to see its results.



Under **Update results**, a summary provides the total number of updates and deployment results on the target VMs. The table on the right shows a detailed breakdown of the updates and the installation results for each.

The available values are:

- **Not attempted** - The update wasn't installed because there was insufficient time available, based on the defined maintenance window duration.
- **Not selected** - The update wasn't selected for deployment.
- **Succeeded** - The update succeeded.
- **Failed** - The update failed.

Select **All logs** to see all log entries that the deployment has created.

Select **Output** to see the job stream of the runbook responsible for managing the update deployment on the target VMs.

Select **Errors** to see detailed information about any errors from the deployment.

Next steps

- To learn how to create alerts to notify you about update deployment results, see [create alerts for Update Management](#).
- To troubleshoot general Update Management errors, see [Troubleshoot Update Management issues](#).

Manage pre-scripts and post-scripts

7/20/2021 • 8 minutes to read • [Edit Online](#)

Pre-scripts and post-scripts are runbooks to run in your Azure Automation account before (pre-task) and after (post-task) an update deployment. Pre-scripts and post-scripts run in the Azure context, not locally. Pre-scripts run at the beginning of the update deployment. Post-scripts run at the end of the deployment and after any reboots that are configured.

Pre-script and post-script requirements

For a runbook to be used as a pre-script or post-script, you must import it into your Automation account and [publish the runbook](#).

Currently, only PowerShell and Python 2 runbooks are supported as Pre/Post scripts. Other runbook types like Python 3, Graphical, PowerShell Workflow, Graphical PowerShell Workflow are currently not supported as Pre/Post scripts.

Pre-script and post-script parameters

When you configure pre-scripts and post-scripts, you can pass in parameters just like scheduling a runbook. Parameters are defined at the time of update deployment creation. Pre-scripts and post-scripts support the following types:

- [char]
- [byte]
- [int]
- [long]
- [decimal]
- [single]
- [double]
- [DateTime]
- [string]

Pre-script and post-script runbook parameters don't support boolean, object, or array types. These values cause the runbooks to fail.

If you need another object type, you can cast it to another type with your own logic in the runbook.

In addition to your standard runbook parameters, the `SoftwareUpdateConfigurationRunContext` parameter (type JSON string) is provided. If you define the parameter in your pre-script or post-script runbook, it's automatically passed in by the update deployment. The parameter contains information about the update deployment, which is a subset of information returned by the [SoftwareUpdateconfigurations API](#). Sections below define the associated properties.

SoftwareUpdateConfigurationRunContext properties

PROPERTY	TYPE	DESCRIPTION
SoftwareUpdateConfigurationName	String	The name of the software update configuration.

PROPERTY	TYPE	DESCRIPTION
SoftwareUpdateConfigurationRunId	GUID	The unique ID for the run.
SoftwareUpdateConfigurationSettings		A collection of properties related to the software update configuration.
SoftwareUpdateConfigurationSettings.OperatingSystem	Int	The operating systems targeted for the update deployment. <code>1</code> = Windows and <code>2</code> = Linux
SoftwareUpdateConfigurationSettings.Duration	Timespan (HH:MM:SS)	The maximum duration of the update deployment run as <code>PT[n]H[n]M[n]S</code> as per ISO8601; also called the maintenance window. Example: 02:00:00
SoftwareUpdateConfigurationSettings.WindowsConfiguration		A collection of properties related to Windows computers.
SoftwareUpdateConfigurationSettings.WindowsConfiguration.excludedKbNumbers	String	A space separated list of KBs that are excluded from the update deployment.
SoftwareUpdateConfigurationSettings.WindowsConfiguration.includedKbNumbers	String	A space separated list of KBs that are included with the update deployment.
SoftwareUpdateConfigurationSettings.WindowsConfiguration.UpdateCategories	Integer	1 = "Critical"; 2 = "Security" 4 = "UpdateRollUp" 8 = "FeaturePack" 16 = "ServicePack" 32 = "Definition" 64 = "Tools" 128 = "Updates"
SoftwareUpdateConfigurationSettings.WindowsConfiguration.rebootSetting	String	Reboot settings for the update deployment. Values are <code>IfRequired</code> , <code>Never</code> , <code>Always</code>
SoftwareUpdateConfigurationSettings.LinuxConfiguration		A collection of properties related to Linux computers.
SoftwareUpdateConfigurationSettings.LinuxConfiguration.IncludedPackageClassifications	Integer	0 = "Unclassified" 1 = "Critical" 2 = "Security" 4 = "Other"
SoftwareUpdateConfigurationSettings.LinuxConfiguration.IncludedPackageNameMasks	String	A space separated list of package names that are included with the update deployment.
SoftwareUpdateConfigurationSettings.LinuxConfiguration.ExcludedPackageNameMasks	String	A space separated list of package names that are excluded from the update deployment.

PROPERTY	TYPE	DESCRIPTION
SoftwareUpdateConfigurationSettings. LinuxConfiguration.RebootSetting	String	Reboot settings for the update deployment. Values are <code>IfRequired</code> , <code>Never</code> , <code>Always</code>
SoftwareUpdateConfigurationSettings.Azur eVirtualMachines	String array	A list of resourceIds for the Azure VMs in the update deployment.
SoftwareUpdateConfigurationSettings.NonAzureComputerNames	String array	A list of the non-Azure computers FQDNs in the update deployment.

The following example is a JSON string passed to the **SoftwareUpdateConfigurationSettings** properties for a Linux computer:

```
"SoftwareUpdateConfigurationSettings": {
    "OperatingSystem": 2,
    "WindowsConfiguration": null,
    "LinuxConfiguration": {
        "IncludedPackageClassifications": 7,
        "ExcludedPackageNameMasks": "fgh xyz",
        "IncludedPackageNameMasks": "abc bin*",
        "RebootSetting": "IfRequired"
    },
    "Targets": {
        "azureQueries": null,
        "nonAzureQueries": ""
    },
    "NonAzureComputerNames": [
        "box1.contoso.com",
        "box2.contoso.com"
    ],
    "AzureVirtualMachines": [
        "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/resourceGroupName/providers/Microsoft.Compute/virtualMachines/vm-01"
    ],
    "Duration": "02:00:00",
    "PSCo mputerName": "localhost",
    "PSShowComputerName": true,
    "PSSourceJobInstanceId": "2477a37b-5262-4f4f-b636-3a70152901e9"
}
```

The following example is a JSON string passed to the **SoftwareUpdateConfigurationSettings** properties for a Windows computer:

```

"SoftwareUpdateConfigurationRunContext": {
    "SoftwareUpdateConfigurationName": "sampleConfiguration",
    "SoftwareUpdateConfigurationRunId": "00000000-0000-0000-0000-000000000000",
    "SoftwareUpdateConfigurationSettings": {
        "operatingSystem": "Windows",
        "duration": "02:00:00",
        "windows": {
            "excludedKbNumbers": [
                "168934",
                "168973"
            ],
            "includedUpdateClassifications": "Critical",
            "rebootSetting": "IfRequired"
        },
        "azureVirtualMachines": [
            "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/myResourceGroup/providers/Microsoft.Compute/virtualMachines/vm-01",
            "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/myResourceGroup/providers/Microsoft.Compute/virtualMachines/vm-02",
            "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/myResourceGroup/providers/Microsoft.Compute/virtualMachines/vm-03"
        ],
        "nonAzureComputerNames": [
            "box1.contoso.com",
            "box2.contoso.com"
        ]
    }
}

```

A full example with all properties can be found at: [Get software update configuration by name](#).

NOTE

The `SoftwareUpdateConfigurationRunContext` object can contain duplicate entries for machines. This can cause pre-scripts and post-scripts to run multiple times on the same machine. To work around this behavior, use `Sort-Object -Unique` to select only unique VM names.

Use a pre-script or post-script in a deployment

To use a pre-script or post-script in an update deployment, start by creating an update deployment. Select **Pre-scripts + Post-Scripts**. This action opens the **Select Pre-scripts + Post-scripts** page.

The screenshot shows two overlapping dialogs. The left dialog is titled 'New update deployment' and contains fields for 'Name' (September Updates), 'Operating system' (Windows selected), 'Items to update' (Add groups and/or machines to update), 'Update classifications' (8 selected), 'Include/exclude updates' (Click to Configure), 'Schedule settings' (Click to Configure), 'Pre-scripts + Post-scripts' (selected), 'Maintenance window (minutes)' (120), and 'Reboot options' (Reboot if required). The right dialog is titled 'Select Pre-scripts + Post-scripts' and lists available items: 'UpdateManagement-SingleHybridWorker' (Published), 'UpdateManagement-TurnOffVms' (Published), and 'UpdateManagement-TurnOnVms' (Published). The 'Selected items' section is currently empty.

Select the script you want to use. In this example, we use the **UpdateManagement-TurnOnVms** runbook. When you select the runbook, the **Configure Script** page opens. Select **Pre-Script**, and then select **OK**.

Repeat this process for the **UpdateManagement-TurnOffVms** script. But when you choose the **Script type**, select **Post-Script**.

The **Selected items** section now shows both your scripts selected. One is a pre-script and the other is a post-script:

Selected items			
NAME	TYPE	PARAMETERS	
UpdateManagement-TurnOnVms	Pre-script	2 configured	...
UpdateManagement-TurnOffVms	Post-script	2 configured	...

Finish configuring your update deployment.

When your update deployment is complete, you can go to **Update deployments** to view the results. As you can see, the status is provided for the pre-script and post-script:

Non-compliant machines: 0 out of 3. Status: Critical and security: 0, Other: 3, Not assessed: 0.

Machines need attention: 3. Status: Critical: 0, Security: 0, Others: 3.

Missing updates: 3.

Failed update deployments: 2 out of 10 in the past six months.

Update management section shows 3 machines: September Updates (Succeeded), New Update Depl... (Succeeded), and June Linux Updates (Succeeded). The table below shows maintenance window utilization and start times.

NAME	STATUS	PRE-SCRIPT STATUS	POST-SCRIPT STATUS	MAINTENANCE WINDOW UTILIZATION	START TIME
September Updates	Succeeded	Completed	Completed	56 / 120 minutes	9/14/2018, 8:37 PM
New Update Depl...	Succeeded	--	--	40 / 120 minutes	7/16/2018, 5:15 PM
June Linux Updates	Succeeded	--	--	4 / 120 minutes	6/14/2018, 12:16 PM

By selecting the update deployment run, you're shown additional details of pre-scripts and post-scripts. A link to the script source at the time of the run is provided.

Overall status: Succeeded

Pre-script status: Completed

Post-script status: Completed

Start time: 9/14/2018, 8:37:58 PM

End time: 9/14/2018, 9:34:03 PM

Pre-script source: [AzureAutomationTutorialScript](#)

Post-script source: [AzureAutomationTutorialScript](#)

Machine update run status: 3 machines. Status breakdown: Failed or Failed to start: 0, In progress: 0, Succeeded: 3, Maintenance window exceeded or too small: 0, Not started: 0.

Updates across machines: 5 Updates. Status breakdown: Failed: 0, Not attempted: 0, Succeeded: 5, Not selected: 0.

MACHINE NAME	STATUS
DC01.internal.lab	Succeeded
SQL01.internal.lab	Succeeded
CAS01.internal.lab	Succeeded

UPDATE NAME	FAILED	NOT ATTEMPTED	NOT SELECTED
2018-09 Cumulative Update for Win...	0 / 3	0 / 3	0 / 3
2018-09 Update for Windows Serve...	0 / 3	0 / 3	0 / 3
Definition Update for Windows Def...	0 / 3	0 / 3	0 / 3
Security Update for SQL Server 201...	0 / 1	0 / 1	0 / 1
Windows Malicious Software Remov...	0 / 3	0 / 3	0 / 3

Stop a deployment

If you want to stop a deployment based on a pre-script, you must **throw** an exception. If you don't, the deployment and post-script will still run. The following code snippet shows how to throw an exception using PowerShell.

```
#In this case, we want to terminate the patch job if any run fails.
#This logic might not hold for all cases - you might want to allow success as long as at least 1 run succeeds
foreach($summary in $finalStatus)
{
    if ($summary.Type -eq "Error")
    {
        #We must throw in order to fail the patch deployment.
        throw $summary.Summary
    }
}
```

In Python 2, exception handling is managed in a **try** block.

Interact with machines

Pre-scripts and post-scripts run as runbooks in your Automation account and not directly on the machines in

your deployment. Pre-tasks and post-tasks also run in the Azure context and don't have access to non-Azure machines. The following sections show how you can interact with the machines directly, whether they're Azure VMs or non-Azure machines.

Interact with Azure machines

Pre-tasks and post-tasks run as runbooks and don't natively run on your Azure VMs in your deployment. To interact with your Azure VMs, you must have the following items:

- A Run As account
- A runbook you want to run

To interact with Azure machines, you should use the [Invoke-AzVMRunCommand](#) cmdlet to interact with your Azure VMs. For an example of how to do this, see the runbook example [Update Management - run script with Run command](#).

Interact with non-Azure machines

Pre-tasks and post-tasks run in the Azure context and don't have access to non-Azure machines. To interact with the non-Azure machines, you must have the following items:

- A Run As account
- Hybrid Runbook Worker installed on the machine
- A runbook you want to run locally
- A parent runbook

To interact with non-Azure machines, a parent runbook is run in the Azure context. This runbook calls a child runbook with the [Start-AzAutomationRunbook](#) cmdlet. You must specify the `RunOn` parameter and provide the name of the Hybrid Runbook Worker for the script to run on. See the runbook example [Update Management - run script locally](#).

Abort patch deployment

If your pre-script returns an error, you might want to abort your deployment. To do that, you must [throw](#) an error in your script for any logic that would constitute a failure.

```
if (<My custom error logic>)
{
    #Throw an error to fail the patch deployment.
    throw "There was an error, abort deployment"
}
```

In Python 2, if you want to throw an error when a certain condition occurs, use a [raise](#) statement.

```
If (<My custom error logic>
    raise Exception('Something happened.')
```

Samples

Samples for pre-scripts and post-scripts can be found in the [Azure Automation GitHub organization](#) and the [PowerShell Gallery](#), or you can import them through the Azure portal. To do that, in your Automation account, under **Process Automation**, select **Runbooks Gallery**. Use **Update Management** for the filter.

The screenshot shows the Azure Automation Runbooks gallery interface. On the left, there's a sidebar with navigation links: Runbooks, Jobs, Runbooks gallery (which is selected), Hybrid worker groups, Watcher tasks, Shared Resources (with Schedules, Modules, and Modules gallery), and a search bar at the top.

The main area displays three runbooks:

- Create a weekly Azure Automation update deployment for one or more VMs in a resource group.** It assumes the VMs are already onboarded to Update Management. Tags: Azure Automation, Azure VM Update Management. Created by: Azure Automation, 265 downloads, Last updated: 6/3/2018.
- Update Management - Turn On VMs** (PowerShell Runbook). This is an example script for Update Management pre/post actions. It starts VMs which aren't currently running to install updates. It can be used in conjunction with UpdateManagement-. Tags: Patching. Created by: Azure Automation, 7 downloads, Last updated: 9/17/2018.
- Update Management - Turn Off VMs** (PowerShell Runbook). This is an example script for Update Management pre/post actions. It stops VMs which were started to install updates. It must be used in conjunction with UpdateManagement-. Tags: Patching. Created by: Azure Automation, 2 downloads, Last updated: 9/17/2018.

Or you can search for them by their script name, as shown in the following list:

- Update Management - Turn On VMs
- Update Management - Turn Off VMs
- Update Management - Run Script Locally
- Update Management - Template for Pre/Post Scripts
- Update Management - Run Script with Run Command

IMPORTANT

After you import the runbooks, you must publish them before they can be used. To do that, find the runbook in your Automation account, select Edit, and then select Publish.

The samples are all based on the basic template that's defined in the following example. This template can be used to create your own runbook to use with pre-scripts and post-scripts. The necessary logic for authenticating with Azure and handling the `SoftwareUpdateConfigurationRunContext` parameter is included.

```

<#
.SYNOPSIS
Barebones script for Update Management Pre/Post

.DESCRIPTION
This script is intended to be run as a part of Update Management pre/post-scripts.
It requires a RunAs account.

.PARAMETER SoftwareUpdateConfigurationRunContext
This is a system variable which is automatically passed in by Update Management during a deployment.
#>

param(
    [string]$SoftwareUpdateConfigurationRunContext
)
#region BoilerplateAuthentication
#This requires a RunAs account
$ServicePrincipalConnection = Get-AutomationConnection -Name 'AzureRunAsConnection'

Add-AzAccount ` 
    -ServicePrincipal ` 
    -TenantId $ServicePrincipalConnection.TenantId ` 
    -ApplicationId $ServicePrincipalConnection.ApplicationId ` 
    -CertificateThumbprint $ServicePrincipalConnection.CertificateThumbprint

$AzureContext = Select-AzSubscription -SubscriptionId $ServicePrincipalConnection.SubscriptionID
#endregion BoilerplateAuthentication

#If you wish to use the run context, it must be converted from JSON
 = ConvertFrom-Json $SoftwareUpdateConfigurationRunContext
#Access the properties of the SoftwareUpdateConfigurationRunContext
$vmIds = $context.SoftwareUpdateConfigurationSettings.AzureVirtualMachines | Sort-Object -Unique
$runId = $context.SoftwareUpdateConfigurationRunId

Write-Output $context

#Example: How to create and write to a variable using the pre-script:
<#
#Create variable named after this run so it can be retrieved
New-AzAutomationVariable -ResourceGroupName $ResourceGroup -AutomationAccountName $AutomationAccount -Name
$runId -Value "" -Encrypted $false
#Set value of variable
Set-AutomationVariable -Name $runId -Value $vmIds
#>

#Example: How to retrieve information from a variable set during the pre-script
<#
$variable = Get-AutomationVariable -Name $runId
#>

```

NOTE

For non-graphical PowerShell runbooks, `Add-AzAccount` and `Add-AzureRMAccount` are aliases for [Connect-AzAccount](#). You can use these cmdlets or you can [update your modules](#) in your Automation account to the latest versions. You might need to update your modules even if you have just created a new Automation account.

Next steps

For details of update management, see [Manage updates and patches for your VMs](#).

How to create alerts for Update Management

3/18/2021 • 2 minutes to read • [Edit Online](#)

Alerts in Azure proactively notify you of results from runbook jobs, service health issues, or other scenarios related to your Automation account. Azure Automation does not include pre-configured alert rules, but you can create your own based on data that it generates. This article provides guidance on creating alert rules using the metrics included with Update Management.

Available metrics

Azure Automation creates two distinct platform metrics related to Update Management that are collected and forwarded to Azure Monitor. These metric are available for analysis using [Metrics Explorer](#) and for alerting using a [metrics alert rule](#).

The two metrics emitted are:

- Total Update Deployment Machine Runs
- Total Update Deployment Runs

When used for alerts, both metrics support dimensions that carry additional information to help scope the alert to a specific update deployment detail. The following table shows the details about the metric and dimensions available when configuring an alert.

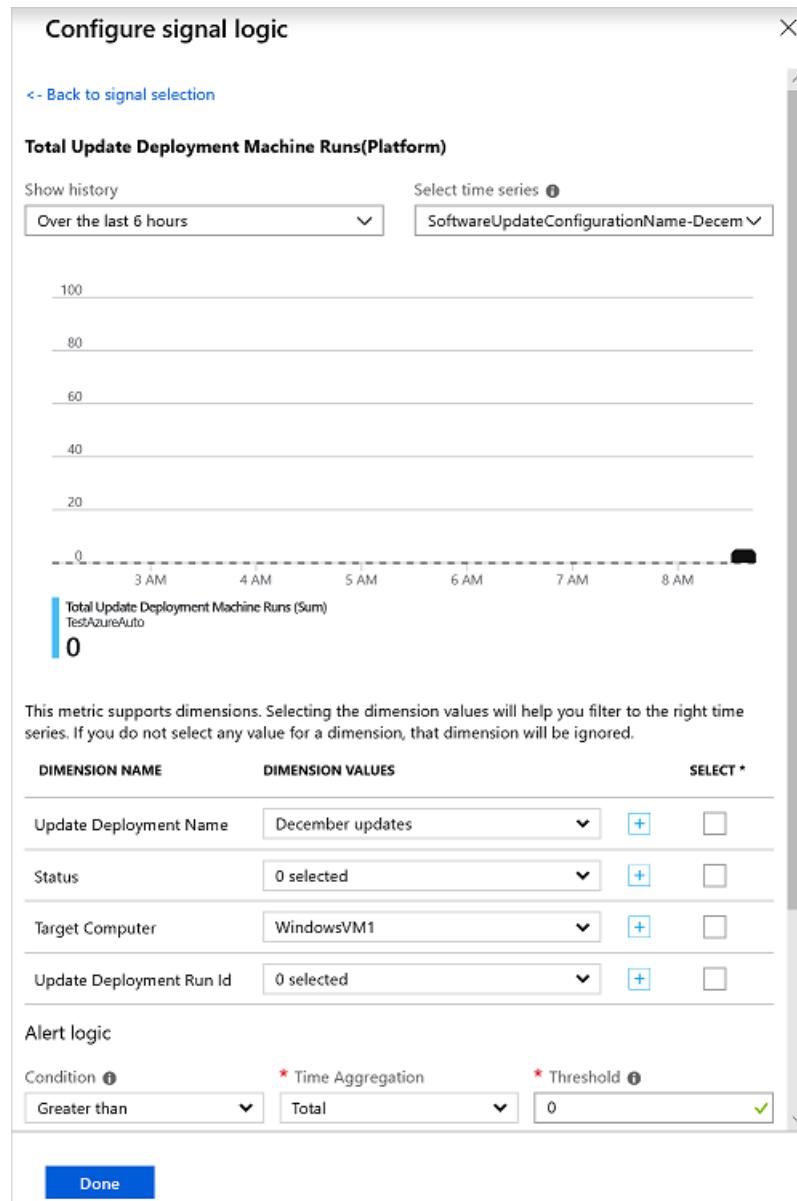
SIGNAL NAME	DIMENSIONS	DESCRIPTION
Total Update Deployment Runs	- Update Deployment Name - Status	Alerts on the overall status of an update deployment.
Total Update Deployment Machine Runs	- Update Deployment Name - Status - Target Computer - Update Deployment Run ID	Alerts on the status of an update deployment targeted at specific machines.

Create alert

Follow the steps below to set up alerts to let you know the status of an update deployment. If you are new to Azure alerts, see [Azure Alerts overview](#).

1. In your Automation account, select **Alerts** under **Monitoring**, and then select **New alert rule**.
2. On the **Create alert rule** page, your Automation account is already selected as the resource. If you want to change it, select **Edit resource**.
3. On the Select a resource page, choose **Automation Accounts** from the **Filter by resource type** dropdown list.
4. Select the Automation account that you want to use, and then select **Done**.
5. Select **Add condition** to chose the signal that's appropriate for your requirement.
6. For a dimension, select a valid value from the list. If the value you want isn't in the list, select + next to the dimension and type in the custom name. Then select the value to look for. If you want to select all values for a dimension, select the **Select *** button. If you don't choose a value for a dimension, Update

Management ignores that dimension.



7. Under **Alert logic**, enter values in the **Time aggregation** and **Threshold** fields, and then select **Done**.
8. On the next page, enter a name and a description for the alert.
9. Set the **Severity** field to **Informational(Sev 2)** for a successful run or **Informational(Sev 1)** for a failed run.

2. Define alert details

* Alert rule name ?
Update Run Succeeded ✓

* Description
A list of computers in the Update Run that were successfully patched. ✓

* Severity ?
Informational(Sev 2) ▼

Enable rule upon creation
 Yes No

Suppress Alerts ?

10. Select **Yes** to enable the alert rule.

Configure action groups for your alerts

Once you have your alerts configured, you can set up an action group, which is a group of actions to use across multiple alerts. The actions can include email notifications, runbooks, webhooks, and much more. To learn more about action groups, see [Create and manage action groups](#).

1. Select an alert and then select **Add action groups** under **Actions**. This will display the **Select an action group to attach to this alert rule** pane.

Select an action group to attach to this alert rule X

The action group selected will attach to this alert rule

[+ Create action group](#)

Subscription (1)
Contoso Suites

Search to filter items...

Action group name	↑↓ Resource group	↑↓ Contain actions
<input type="checkbox"/> Application Insights Smart Detection	MAIC-RG	2 Email Azure Resource Manager Roles (1)
<input type="checkbox"/> StSt_AutoStop_AG_ARM-MAIC-RG	MAIC-RG	1 Webhook (1)
<input type="checkbox"/> StartStop_VM_Notification	prod1	1 Email (1)

Select

2. Select the checkbox for the Action group to attach and press Select.

Next steps

- Learn more about [alerts in Azure Monitor](#).
- Learn about [log queries](#) to retrieve and analyze data from a Log Analytics workspace.
- Manage [usage and costs with Azure Monitor Logs](#) describes how to control your costs by changing your data retention period, and how to analyze and alert on your data usage.

Integrate Update Management with Microsoft Endpoint Configuration Manager

7/14/2021 • 3 minutes to read • [Edit Online](#)

Customers who have invested in Microsoft Endpoint Configuration Manager to manage PCs, servers, and mobile devices also rely on its strength and maturity in managing software updates as part of their software update management (SUM) cycle.

You can report and update managed Windows servers by creating and pre-staging software update deployments in Microsoft Endpoint Configuration Manager, and get detailed status of completed update deployments using [Update Management](#). If you use Microsoft Endpoint Configuration Manager for update compliance reporting, but not for managing update deployments with your Windows servers, you can continue reporting to Microsoft Endpoint Configuration Manager while security updates are managed with Azure Automation Update Management.

NOTE

While Update Management supports update assessment and patching of Windows Server 2008 R2, it does not support clients managed by Microsoft Endpoint Configuration Manager running this operating system.

Prerequisites

- You must have [Azure Automation Update Management](#) added to your Automation account.
- Windows servers currently managed by your Microsoft Endpoint Configuration Manager environment also need to report to the Log Analytics workspace that also has Update Management enabled.
- This feature is enabled in Microsoft Endpoint Configuration Manager current branch version 1606 and higher. To integrate your Microsoft Endpoint Configuration Manager central administration site or a standalone primary site with Azure Monitor logs and import collections, review [Connect Configuration Manager to Azure Monitor logs](#).
- Windows agents must either be configured to communicate with a Windows Server Update Services (WSUS) server or have access to Microsoft Update if they don't receive security updates from Microsoft Endpoint Configuration Manager.

How you manage clients hosted in Azure IaaS with your existing Microsoft Endpoint Configuration Manager environment primarily depends on the connection you have between Azure datacenters and your infrastructure. This connection affects any design changes you may need to make to your Microsoft Endpoint Configuration Manager infrastructure and related cost to support those necessary changes. To understand what planning considerations you need to evaluate before proceeding, review [Configuration Manager on Azure - Frequently Asked Questions](#).

Manage software updates from Microsoft Endpoint Configuration Manager

Perform the following steps if you are going to continue managing update deployments from Microsoft Endpoint Configuration Manager. Azure Automation connects to Microsoft Endpoint Configuration Manager to apply updates to the client computers connected to your Log Analytics workspace. Update content is available from the client computer cache as if the deployment were managed by Microsoft Endpoint Configuration Manager.

1. Create a software update deployment from the top-level site in your Microsoft Endpoint Configuration Manager hierarchy using the process described in [Deploy software updates](#). The only setting that must be configured differently from a standard deployment is the **Installation deadline** option in Endpoint Configuration Manager. It needs to be set to a future date to ensure only Automation Update Management initiates the update deployment. This setting is described under [Step 4, Deploy the software update group](#).
2. In Endpoint Configuration Manager, configure the **User notifications** option to prevent displaying notifications on the target machines. We recommend setting the **Hide in Software Center and all notifications** option to avoid a logged on user from being notified of a scheduled update deployment and manually deploying those updates. This setting is described under [Step 4, Deploy the software update group](#).
3. In Azure Automation, select **Update Management**. Create a new deployment following the steps described in [Creating an Update Deployment](#) and select **Imported groups** on the **Type** dropdown to select the appropriate Microsoft Endpoint Configuration Manager collection. Keep in mind the following important points:
 - a. If a maintenance window is defined on the selected Microsoft Endpoint Configuration Manager device collection, members of the collection honor it instead of the **Duration** setting defined in the scheduled deployment.
 - b. Members of the target collection must have a connection to the Internet (either direct, through a proxy server or through the Log Analytics gateway).

After completing the update deployment through Azure Automation, the target computers that are members of the selected computer group will install updates at the scheduled time from their local client cache. You can [view update deployment status](#) to monitor the results of your deployment.

Manage software updates from Azure Automation

To manage updates for Windows Server VMs that are Microsoft Endpoint Configuration Manager clients, you need to configure client policy to disable the Software Update Management feature for all clients managed by Update Management. By default, client settings target all devices in the hierarchy. For more information about this policy setting and how to configure it, review [How to configure client settings in Configuration Manager](#).

After performing this configuration change, you create a new deployment following the steps described in [Creating an Update Deployment](#) and select **Imported groups** on the **Type** drop-down to select the appropriate Microsoft Endpoint Configuration Manager collection.

Next steps

To set up an integration schedule, see [Schedule an update deployment](#).

Configure Windows Update settings for Azure Automation Update Management

11/2/2020 • 2 minutes to read • [Edit Online](#)

Azure Automation Update Management relies on the [Windows Update client](#) to download and install Windows updates. There are specific settings that are used by the Windows Update client when connecting to Windows Server Update Services (WSUS) or Windows Update. Many of these settings can be managed with:

- Local Group Policy Editor
- Group Policy
- PowerShell
- Directly editing the Registry

Update Management respects many of the settings specified to control the Windows Update client. If you use settings to enable non-Windows updates, Update Management will also manage those updates. If you want to enable downloading of updates before an update deployment occurs, update deployment can be faster, more efficient, and less likely to exceed the maintenance window.

For additional recommendations on setting up WSUS in your Azure subscription and securely keep your Windows virtual machines up to date, review [Plan your deployment for updating Windows virtual machines in Azure using WSUS](#).

Pre-download updates

To configure the automatic downloading of updates without automatically installing them, you can use Group Policy to [configure the Automatic Updates setting](#) to 3. This setting enables downloads of the required updates in the background, and notifies you that the updates are ready to install. In this way, Update Management remains in control of schedules, but allows downloading of updates outside the Update Management maintenance window. This behavior prevents `Maintenance window exceeded` errors in Update Management.

You can enable this setting in PowerShell:

```
$WUSettings = (New-Object -com "Microsoft.Update.AutoUpdate").Settings  
$WUSettings.NotificationLevel = 3  
$WUSettings.Save()
```

Configure reboot settings

The registry keys listed in [Configuring Automatic Updates by editing the registry](#) and [Registry keys used to manage restart](#) can cause your machines to reboot, even if you specify **Never Reboot** in the **Update Deployment** settings. Configure these registry keys to best suit your environment.

Enable updates for other Microsoft products

By default, the Windows Update client is configured to provide updates only for Windows. If you enable the **Give me updates for other Microsoft products when I update Windows** setting, you also receive updates for other products, including security patches for Microsoft SQL Server and other Microsoft software. You can configure this option if you have downloaded and copied the latest [Administrative template files](#) available for Windows 2016 and later.

If you have machines running Windows Server 2012 R2, you can't configure this setting through Group Policy. Run the following PowerShell command on these machines:

```
$ServiceManager = (New-Object -com "Microsoft.Update.ServiceManager")
$ServiceManager.Services
$ServiceID = "7971f918-a847-4430-9279-4a52d1efe18d"
$ServiceManager.AddService2($ServiceId,7,"")
```

Make WSUS configuration settings

Update Management supports WSUS settings. You can specify sources for scanning and downloading updates using instructions in [Specify intranet Microsoft Update service location](#). By default, the Windows Update client is configured to download updates from Windows Update. When you specify a WSUS server as a source for your machines, if the updates aren't approved in WSUS, update deployment fails.

To restrict machines to the internal update service, set [Do not connect to any Windows Update Internet locations](#).

Next steps

Schedule an update deployment by following instructions in [Manage updates and patches for your VMs](#).

Use dynamic groups with Update Management

6/22/2021 • 2 minutes to read • [Edit Online](#)

Update Management allows you to target a dynamic group of Azure or non-Azure VMs for update deployments. Using a dynamic group keeps you from having to edit your deployment to update machines.

NOTE

Dynamic groups do not work with classic VMs.

You can define dynamic groups for Azure or non-Azure machines from **Update management** in the Azure portal. See [Manage updates for VMs](#).

A dynamic group is defined by a query that Azure Automation evaluates at deployment time. Even if a dynamic group query retrieves a large number of machines, Azure Automation can process only a maximum of 1000 machines at a time. See [Azure subscription and service limits, quotas, and constraints](#).

NOTE

If you expect to update more than 1000 machines, we recommend that you split up the updates among multiple update schedules.

Define dynamic groups for Azure machines

When defining a dynamic group query for Azure machines, you can use the following items to populate the dynamic group:

- Subscription
- Resource groups
- Locations
- Tags

To preview the results of your dynamic group query, click **Preview**. The preview shows the group membership at the current time. In the example, we're searching for machines having the tag `Role` for the group `BackendServer`. If more machines have this tag added, they are added to any future deployments against that group.

name	subscription	resource group	location
DC02	Parnell Aerospace	MAIC-RG	East US 2
SVR01	Parnell Aerospace	MAIC-RG	East US 2
CMPRI01	Parnell Aerospace	maic-rg	East US 2

Define dynamic groups for non-Azure machines

A dynamic group for non-Azure machines uses saved searches, also called computer groups. To learn how to create a saved search, see [Creating a computer group](#). Once your saved search is created, you can select it from the list of saved searches in **Update management** in the Azure portal. Click **Preview** to preview the computers in the saved search.

The screenshot shows the 'Select groups' dialog box within the 'New update deployment' interface. On the left, there's a sidebar with various configuration options like 'Name', 'Operating system', and 'Groups to update'. The main area is titled 'Select groups' and contains sections for 'Included items' and 'Available Items'. The 'Included items' section lists 'OnPrem' under 'NAME' and 'groups' under 'TYPE'. The 'Available Items' section lists 'OnPrem1', 'OnPrem2', and 'OnPrem3'. A note at the top right states: 'The final list of machines to be updated is evaluated at deployment time and may differ from the list below. Additionally, only non-Azure machines will appear in the list.'

NOTE

A saved search that [queries data stored across multiple Log Analytics workspaces](#) is not supported.

Next steps

You can [query Azure Monitor logs](#) to analyze update assessments, deployments, and other related management tasks. It includes pre-defined queries to help you get started.

Query Update Management logs

8/13/2021 • 14 minutes to read • [Edit Online](#)

In addition to the details that are provided during Update Management deployment, you can search the logs stored in your Log Analytics workspace. To search the logs from your Automation account, select **Update management** and open the Log Analytics workspace associated with your deployment.

You can also customize the log queries or use them from different clients. See [Log Analytics search API documentation](#).

Query update records

Update Management collects records for Windows and Linux VMs and the data types that appear in log search results. The following sections describe those records.

Query required updates

A record with a type of `RequiredUpdate` is created that represents updates required by a machine. These records have the properties in the following table:

PROPERTY	DESCRIPTION
Computer	Fully-qualified domain name of reporting machine.
KBID	Knowledge base article ID for the Windows update.
ManagementGroupName	Name of the Operations Manager management group or Log Analytics workspace.
Product	The products for which the update is applicable for.
PublishDate	The date the update is ready to be downloaded and installed from Windows Update.
Server	
SourceHealthServiceId	Unique identifier representing the Log Analytics Windows agent ID.
SourceSystem	<i>OperationsManager</i>
TenantId	Unique identifier representing your organization's instance of Azure Active Directory.
TimeGenerated	Date and time that the record was created.
Type	<i>Update</i>

PROPERTY	DESCRIPTION
UpdateClassification	Indicates the type of updates that can be applied. For Windows: <i>Critical updates</i> <i>Security updates</i> <i>Update rollups</i> <i>Feature packs</i> <i>Service packs</i> <i>Definition updates</i> <i>Tools</i> Updates. For Linux: <i>Critical and security updates</i> <i>Other</i>
UpdateSeverity	Severity rating for the vulnerability. Values are: <i>Critical</i> <i>Important</i> <i>Moderate</i> <i>Low</i>
UpdateTitle	The title of the update.

Query Update record

A record with a type of `Update` is created that represents updates available and their installation status for a machine. These records have the properties in the following table:

PROPERTY	DESCRIPTION
ApprovalSource	Applies to Windows operating system only. Source of approval for the record. The value is Microsoft Update.
Approved	True if the record is approved, or False otherwise.
Classification	Approval classification. The value is Updates.
Computer	Fully-qualified domain name of reporting machine.
ComputerEnvironment	Environment. Possible values are Azure or Non-Azure.
MSRCBulletinID	Security bulletin ID number.
MSRCSecurity	Severity rating for the vulnerability. Values are: <i>Critical</i> <i>Important</i> <i>Moderate</i> <i>Low</i>
KBID	Knowledge base article ID for the Windows update.
ManagementGroupName	Name of the Operations Manager management group or the Log Analytics workspace.
UpdateID	Unique identifier of the software update.
RevisionNumber	The revision number of a specific revision of an update.

PROPERTY	DESCRIPTION
Optional	True if the record is optional, or False otherwise.
RebootBehavior	The reboot behavior after installing/uninstalling an update.
_ResourceId	Unique identifier for the resource associated with the record.
Type	Record type. The value is Update.
VMUUID	Unique identifier for the virtual machine.
MG	Unique identifier for the management group or Log Analytics workspace.
TenantId	Unique identifier representing your organization's instance of Azure Active Directory.
SourceSystem	The source system for the record. The value is OperationsManager .
TimeGenerated	Date and time of record creation.
SourceComputerId	Unique identifier representing the source computer.
Title	The title of the update.
PublishedDate (UTC)	The date when the update is ready to be downloaded and installed from Windows Update.
UpdateState	The current state of the update.
Product	The products for which the update is applicable.
SubscriptionId	Unique identifier for the Azure subscription.
ResourceGroup	Name of the resource group to which the resource belongs.
ResourceProvider	The resource provider.
Resource	Name of the resource.
ResourceType	The resource type.

Query Update Agent record

A record with a type of `UpdateAgent` is created that provides details of the update agent on the machine. These records have the properties in the following table:

PROPERTY	DESCRIPTION
AgeofOldestMissingRequiredUpdate	
AutomaticUpdateEnabled	

PROPERTY	DESCRIPTION
Computer	Fully-qualified domain name of reporting machine.
DaySinceLastUpdateBucket	
ManagementGroupName	Name of the Operations Manager management group or Log Analytics workspace.
OSVersion	The version of the operating system.
Server	
SourceHealthServiceId	Unique identifier representing the Log Analytics Windows agent ID.
SourceSystem	The source system for the record. The value is <code>OperationsManager</code> .
TenantId	Unique identifier representing your organization's instance of Azure Active Directory.
TimeGenerated	Date and time of record creation.
Type	Record type. The value is <code>Update</code> .
WindowsUpdateAgentVersion	Version of the Windows Update agent.
WSUSServer	Errors if the Windows Update agent has a problem, to assist with troubleshooting.

Query Update Deployment Status record

A record with a type of `UpdateRunProgress` is created that provides update deployment status of a scheduled deployment by machine. These records have the properties in the following table:

PROPERTY	DESCRIPTION
Computer	Fully-qualified domain name of reporting machine.
ComputerEnvironment	Environment. Values are Azure or Non-Azure.
CorrelationId	Unique identifier of the runbook job run for the update.
EndTime	The time when the synchronization process ended. <i>This property is currently not used. See TimeGenerated.</i>
ErrorResult	Windows Update error code generated if an update fails to install.

PROPERTY	DESCRIPTION
InstallationStatus	<p>The possible installation states of an update on the client computer,</p> <ul style="list-style-type: none"> <code>NotStarted</code> - job not triggered yet. <code>FailedToStart</code> - unable to start the job on machine. <code>Failed</code> - job started but failed with an exception. <code>InProgress</code> - job in progress. <code>MaintenanceWindowExceeded</code> - if execution was remaining but maintenance window interval reached. <code>Succeeded</code> - job succeeded. <code>InstallFailed</code> - update failed to install successfully. <code>NotIncluded</code> <code>Excluded</code>
KBID	Knowledge base article ID for the Windows update.
ManagementGroupName	Name of the Operations Manager management group or Log Analytics workspace.
OSType	Type of operating system. Values are Windows or Linux.
Product	The products for which the update is applicable.
Resource	Name of the resource.
ResourceId	Unique identifier for the resource associated with the record.
ResourceProvider	The resource provider.
ResourceType	Resource type.
SourceComputerId	Unique identifier representing the source computer.
SourceSystem	Source system for the record. The value is <code>OperationsManager</code> .
StartTime	Time when the update is scheduled to be installed. <i>This property is currently not used. See TimeGenerated.</i>
SubscriptionId	Unique identifier for the Azure subscription.
SucceededOnRetry	Value indicating if the update execution failed on the first attempt and the current operation is a retry attempt.
TimeGenerated	Date and time of record creation.
Title	The title of the update.
Type	The type of update. The value is <code>UpdateRunProgress</code> .
UpdatedId	Unique identifier of the software update.
VMUUID	Unique identifier for the virtual machine.

PROPERTY	DESCRIPTION
ResourceId	Unique identifier for the resource associated with the record.

Query Update Summary record

A record with a type of `UpdateSummary` is created that provides update summary by machine. These records have the properties in the following table:

PROPERTY	DESCRIPTION
Computer	Fully-qualified domain name of reporting machine.
ComputerEnvironment	Environment. Values are Azure or Non-Azure.
CriticalUpdatesMissing	Number of applicable critical updates that are missing.
ManagementGroupName	Name of the Operations Manager management group or Log Analytics workspace.
.NETRuntimeVersion	Version of .NET Framework installed on the Windows computer.
OldestMissingSecurityUpdateBucket	Specifier of the oldest missing security bucket. Values are: Recent if value is less than 30 days 30 days ago 60 days ago 90 days ago 120 days ago 150 days ago 180 days ago Older when value is greater than 180 days.
OldestMissingSecurityUpdateInDays	Total number of days for the oldest update detected as applicable that has not been installed.
OsVersion	The version of the operating system.
OtherUpdatesMissing	Count of detected updates missing.
Resource	Name of the resource for the record.
ResourceGroup	Name of the resource group containing the resource.
ResourceId	Unique identifier for the resource associated with the record.
ResourceProvider	The resource provider.
ResourceType	Resource type.
RestartPending	True if a restart is pending, or False otherwise.
SecurityUpdatesMissing	Count of missing security updates that are applicable.
SourceComputerId	Unique identifier for the virtual machine.

PROPERTY	DESCRIPTION
SourceSystem	Source system for the record. The value is <code>OpsManager</code> .
SubscriptionId	Unique identifier for the Azure subscription.
TimeGenerated	Date and time of record creation.
TotalUpdatesMissing	Total number of missing updates that are applicable.
Type	Record type. The value is <code>UpdateSummary</code> .
VMUUID	Unique identifier for the virtual machine.
WindowsUpdateAgentVersion	Version of the Windows Update agent.
WindowsUpdateSetting	Status of the Windows Update agent. Possible values are: <code>Scheduled installation</code> <code>Notify before installation</code> <code>Error returned from unhealthy WUA agent</code>
WSUSServer	Errors if the Windows Update agent has a problem, to assist with troubleshooting.
_ResourceId	Unique identifier for the resource associated with the record.

Sample queries

The following sections provide sample log queries for update records that are collected for Update Management.

Confirm that non-Azure machines are enabled for Update Management

To confirm that directly connected machines are communicating with Azure Monitor logs, run one of the following log searches.

Linux

```
Heartbeat
| where OSType == "Linux" | summarize arg_max(TimeGenerated, *) by SourceComputerId | top 500000 by Computer
asc | render table
```

Windows

```
Heartbeat
| where OSType == "Windows" | summarize arg_max(TimeGenerated, *) by SourceComputerId | top 500000 by Computer
asc | render table
```

On a Windows computer, you can review the following information to verify agent connectivity with Azure Monitor logs:

1. In Control Panel, open **Microsoft Monitoring Agent**. On the Azure Log Analytics tab, the agent displays the following message: **The Microsoft Monitoring Agent has successfully connected to Log Analytics**.
2. Open the Windows Event Log. Go to Application and Services Logs\Operations Manager and

search for Event ID 3000 and Event ID 5002 from the source **Service Connector**. These events indicate that the computer has registered with the Log Analytics workspace and is receiving configuration.

If the agent can't communicate with Azure Monitor logs and the agent is configured to communicate with the internet through a firewall or proxy server, confirm the firewall or proxy server is properly configured. To learn how to verify the firewall or proxy server is properly configured, see [Network configuration for Windows agent](#) or [Network configuration for Linux agent](#).

NOTE

If your Linux systems are configured to communicate with a proxy or Log Analytics Gateway and you're enabling Update Management, update the `proxy.conf` permissions to grant the `omiuser` group read permission on the file by using the following commands:

```
sudo chown omsagent:omiusers /etc/opt/microsoft/omsagent/proxy.conf  
sudo chmod 644 /etc/opt/microsoft/omsagent/proxy.conf
```

Newly added Linux agents show a status of **Updated** after an assessment has been performed. This process can take up to 6 hours.

To confirm that an Operations Manager management group is communicating with Azure Monitor logs, see [Validate Operations Manager integration with Azure Monitor logs](#).

Single Azure VM Assessment queries (Windows)

Replace the `VMUUID` value with the VM GUID of the virtual machine you're querying. You can find the `VMUUID` that should be used by running the following query in Azure Monitor logs:

```
Update | where Computer == "<machine name>" | summarize by Computer, VMUUID
```

Missing updates summary

```
Update  
| where TimeGenerated>ago(14h) and OSType!="Linux" and (Optional==false or Classification has "Critical" or  
Classification has "Security") and VMUUID=~"b08d5afa-1471-4b52-bd95-a44fea6e4ca8"  
| summarize hint.strategy=partitioned arg_max(TimeGenerated, UpdateState, Classification, Approved) by  
Computer, SourceComputerId, UpdateID  
| where UpdateState=~"Needed" and Approved!=false  
| summarize by UpdateID, Classification  
| summarize allUpdatesCount=count(), criticalUpdatesCount=countif(Classification has "Critical"),  
securityUpdatesCount=countif(Classification has "Security"), otherUpdatesCount=countif(Classification !has  
"Critical" and Classification !has "Security")
```

Missing updates list

```
Update  
| where TimeGenerated>ago(14h) and OSType!="Linux" and (Optional==false or Classification has "Critical" or  
Classification has "Security") and VMUUID=~"8bf1cccc6-b6d3-4a0b-a643-23f346dfdf82"  
| summarize hint.strategy=partitioned arg_max(TimeGenerated, UpdateState, Classification, Title, KBID,  
PublishedDate, Approved) by Computer, SourceComputerId, UpdateID  
| where UpdateState=~"Needed" and Approved!=false  
| project-away UpdateState, Approved, TimeGenerated  
| summarize computersCount=dcount(SourceComputerId, 2), displayName=any(Title),  
publishedDate=min(PublishedDate), ClassificationWeight=max(if(Classification has "Critical", 4,  
iff(Classification has "Security", 2, 1))) by id=strcat(UpdateID, "_", KBID), classification=Classification,  
InformationId=strcat("KB", KBID), InformationUrl=iff(isnotempty(KBID),  
strcat("https://support.microsoft.com/kb/", KBID), ""), osType=2  
| sort by ClassificationWeight desc, computersCount desc, displayName asc  
| extend informationLink=(iff(isnotempty(InformationId) and isnotempty(InformationUrl), toobject(strcat('{  
"uri": "", InformationUrl, "text": "", InformationId, "target": "blank" }'))), toobject(''))  
| project-away ClassificationWeight, InformationId, InformationUrl
```

Single Azure VM assessment queries (Linux)

For some Linux distros, there is an [endianness](#) mismatch with the VMUUID value that comes from Azure Resource Manager and what is stored in Azure Monitor logs. The following query checks for a match on either endianness. Replace the VMUUID values with the big-endian and little-endian format of the GUID to properly return the results. You can find the VMUUID that should be used by running the following query in Azure Monitor logs:

```
Update | where Computer == "<machine name>" | summarize by Computer, VMUUID
```

Missing updates summary

```
Update
| where TimeGenerated>ago(5h) and OSType=="Linux" and (VMUUID=~"625686a0-6d08-4810-aae9-a089e68d4911" or
VMUUID=~"a0865662-086d-1048-aae9-a089e68d4911")
| summarize hint.strategy=partitioned arg_max(TimeGenerated, UpdateState, Classification) by Computer,
SourceComputerId, Product, ProductArch
| where UpdateState=~"Needed"
| summarize by Product, ProductArch, Classification
| summarize allUpdatesCount=count(), criticalUpdatesCount=countif(Classification has "Critical"),
securityUpdatesCount=countif(Classification has "Security"), otherUpdatesCount=countif(Classification !has
"Critical" and Classification !has "Security")
```

Missing updates list

```
Update
| where TimeGenerated>ago(5h) and OSType=="Linux" and (VMUUID=~"625686a0-6d08-4810-aae9-a089e68d4911" or
VMUUID=~"a0865662-086d-1048-aae9-a089e68d4911")
| summarize hint.strategy=partitioned arg_max(TimeGenerated, UpdateState, Classification, BulletinUrl,
BulletinID) by Computer, SourceComputerId, Product, ProductArch
| where UpdateState=~"Needed"
| project-away UpdateState, TimeGenerated
| summarize computersCount=dcount(SourceComputerId, 2), ClassificationWeight=max(if(Classification has
"Critical", 4, iff(Classification has "Security", 2, 1))) by id=strcat(Product, "_", ProductArch),
displayName=Product, productArch=ProductArch, classification=Classification, InformationId=BulletinID,
InformationUrl=tostring(split(BulletinUrl, ";", 0)[0]), osType=1
| sort by ClassificationWeight desc, computersCount desc, displayName asc
| extend informationLink=(iff(isnotempty(InformationId) and isnotempty(InformationUrl), toobject(strcat('{",
"uri": "", InformationUrl, '", "text": "", InformationId, '", "target": "blank" }'))), toobject(''))
| project-away ClassificationWeight, InformationId, InformationUrl
```

Multi-VM assessment queries

Computers summary

```

Heartbeat
| where TimeGenerated>ago(12h) and OSType=~"Windows" and notempty(Computer)
| summarize arg_max(TimeGenerated, Solutions) by SourceComputerId
| where Solutions has "updates"
| distinct SourceComputerId
| join kind=leftouter
(
    Update
    | where TimeGenerated>ago(14h) and OSType!="Linux"
    | summarize hint.strategy=partitioned arg_max(TimeGenerated, UpdateState, Approved, Optional,
Classification) by SourceComputerId, UpdateID
        | distinct SourceComputerId, Classification, UpdateState, Approved, Optional
        | summarize WorstMissingUpdateSeverity=max(if(UpdateState=~"Needed" and (Optional==false or
Classification has "Critical" or Classification has "Security") and Approved!=false, iff(Classification has
"Critical", 4, iff(Classification has "Security", 2, 1)), 0)) by SourceComputerId
    )
on SourceComputerId
| extend WorstMissingUpdateSeverity=coalesce(WorstMissingUpdateSeverity, -1)
| summarize computersBySeverity=count() by WorstMissingUpdateSeverity
| union (Heartbeat
| where TimeGenerated>ago(12h) and OSType=="Linux" and notempty(Computer)
| summarize arg_max(TimeGenerated, Solutions) by SourceComputerId
| where Solutions has "updates"
| distinct SourceComputerId
| join kind=leftouter
(
    Update
    | where TimeGenerated>ago(5h) and OSType=="Linux"
    | summarize hint.strategy=partitioned arg_max(TimeGenerated, UpdateState, Classification) by
SourceComputerId, Product, ProductArch
        | distinct SourceComputerId, Classification, UpdateState
        | summarize WorstMissingUpdateSeverity=max(if(UpdateState=~"Needed", iff(Classification has "Critical",
4, iff(Classification has "Security", 2, 1)), 0)) by SourceComputerId
    )
on SourceComputerId
| extend WorstMissingUpdateSeverity=coalesce(WorstMissingUpdateSeverity, -1)
| summarize computersBySeverity=count() by WorstMissingUpdateSeverity
| summarize assessedComputersCount=sumif(computersBySeverity, WorstMissingUpdateSeverity>-1),
notAssessedComputersCount=sumif(computersBySeverity, WorstMissingUpdateSeverity==1),
computersNeedCriticalUpdatesCount=sumif(computersBySeverity, WorstMissingUpdateSeverity==4),
computersNeedSecurityUpdatesCount=sumif(computersBySeverity, WorstMissingUpdateSeverity==2),
computersNeedOtherUpdatesCount=sumif(computersBySeverity, WorstMissingUpdateSeverity==1),
upToDateComputersCount=sumif(computersBySeverity, WorstMissingUpdateSeverity==0)
| summarize assessedComputersCount=sum(assessedComputersCount),
computersNeedCriticalUpdatesCount=sum(computersNeedCriticalUpdatesCount),
computersNeedSecurityUpdatesCount=sum(computersNeedSecurityUpdatesCount),
computersNeedOtherUpdatesCount=sum(computersNeedOtherUpdatesCount),
upToDateComputersCount=sum(upToDateComputersCount), notAssessedComputersCount=sum(notAssessedComputersCount)
| extend allComputersCount=assessedComputersCount+notAssessedComputersCount

```

Missing updates summary

```

Update
| where TimeGenerated>ago(5h) and OSType=="Linux" and SourceComputerId in ((Heartbeat
| where TimeGenerated>ago(12h) and OSType=="Linux" and notempty(Computer)
| summarize arg_max(TimeGenerated, Solutions) by SourceComputerId
| where Solutions has "updates"
| distinct SourceComputerId)
| summarize hint.strategy=partitioned arg_max(TimeGenerated, UpdateState, Classification) by Computer,
SourceComputerId, Product, ProductArch
| where UpdateState=~"Needed"
| summarize by Product, ProductArch, Classification
| union (Update
| where TimeGenerated>ago(14h) and OSType!="Linux" and (Optional==false or Classification has "Critical" or
Classification has "Security") and SourceComputerId in ((Heartbeat
| where TimeGenerated>ago(12h) and OSType=~"Windows" and notempty(Computer)
| summarize arg_max(TimeGenerated, Solutions) by SourceComputerId
| where Solutions has "updates"
| distinct SourceComputerId)
| summarize hint.strategy=partitioned arg_max(TimeGenerated, UpdateState, Classification, Approved) by
Computer, SourceComputerId, UpdateID
| where UpdateState=~"Needed" and Approved!=false
| summarize by UpdateID, Classification )
| summarize allUpdatesCount=count(), criticalUpdatesCount=countif(Classification has "Critical"),
securityUpdatesCount=countif(Classification has "Security"), otherUpdatesCount=countif(Classification !has
"Critical" and Classification !has "Security")

```

Computers list

```

Heartbeat
| where TimeGenerated>ago(12h) and OSType=="Linux" and notempty(Computer)
| summarize arg_max(TimeGenerated, Solutions, Computer, ResourceId, ComputerEnvironment, VMUUID) by
SourceComputerId
| where Solutions has "updates"
| extend vmuuid=VMUUID, azureResourceId=ResourceId, osType=1, environment=iff(ComputerEnvironment=~"Azure",
1, 2), scopedToUpdatesSolution=true, lastUpdateAgentSeenTime=""
| join kind=leftouter
(
    Update
    | where TimeGenerated>ago(5h) and OSType=="Linux" and SourceComputerId in ((Heartbeat
    | where TimeGenerated>ago(12h) and OSType=="Linux" and notempty(Computer)
    | summarize arg_max(TimeGenerated, Solutions) by SourceComputerId
    | where Solutions has "updates"
    | distinct SourceComputerId)
    | summarize hint.strategy=partitioned arg_max(TimeGenerated, UpdateState, Classification, Product,
Computer, ComputerEnvironment) by SourceComputerId, Product, ProductArch
        | summarize Computer=any(Computer), ComputerEnvironment=any(ComputerEnvironment),
missingCriticalUpdatesCount=countif(Classification has "Critical" and UpdateState=~"Needed"),
missingSecurityUpdatesCount=countif(Classification has "Security" and UpdateState=~"Needed"),
missingOtherUpdatesCount=countif(Classification !has "Critical" and Classification !has "Security" and
UpdateState=~"Needed"), lastAssessedTime=max(TimeGenerated), lastUpdateAgentSeenTime="" by SourceComputerId
            | extend compliance=iff(missingCriticalUpdatesCount > 0 or missingSecurityUpdatesCount > 0, 2, 1)
            | extend ComplianceOrder=iff(missingCriticalUpdatesCount > 0 or missingSecurityUpdatesCount > 0 or
missingOtherUpdatesCount > 0, 1, 3)
    )
on SourceComputerId
| project id=SourceComputerId, displayName=Computer, sourceComputerId=SourceComputerId,
scopedToUpdatesSolution=true, missingCriticalUpdatesCount=coalesce(missingCriticalUpdatesCount, -1),
missingSecurityUpdatesCount=coalesce(missingSecurityUpdatesCount, -1),
missingOtherUpdatesCount=coalesce(missingOtherUpdatesCount, -1), compliance=coalesce(compliance, 4),
lastAssessedTime, lastUpdateAgentSeenTime, osType=1, environment=iff(ComputerEnvironment=~"Azure", 1, 2),
ComplianceOrder=coalesce(ComplianceOrder, 2)
| union(Heartbeat
| where TimeGenerated>ago(12h) and OSType=~"Windows" and notempty(Computer)
| summarize arg_max(TimeGenerated, Solutions, Computer, ResourceId, ComputerEnvironment, VMUUID) by
SourceComputerId
| where Solutions has "updates"
| extend vmuuid=VMUUID, azureResourceId=ResourceId, osType=2, environment=iff(ComputerEnvironment=~"Azure",
1, 2), scopedToUpdatesSolution=true, lastUpdateAgentSeenTime=""

```

```

1, 2), scopedToUpdatesSolution=true, lastUpdateAgentSeenTime-
| join kind=leftouter
(
    Update
    | where TimeGenerated>ago(14h) and OSType!="Linux" and SourceComputerId in ((Heartbeat
    | where TimeGenerated>ago(12h) and OSType=~"Windows" and notempty(Computer)
    | summarize arg_max(TimeGenerated, Solutions) by SourceComputerId
    | where Solutions has "updates"
    | distinct SourceComputerId)
    | summarize hint.strategy=partitioned arg_max(TimeGenerated, UpdateState, Classification, Title,
Optional, Approved, Computer, ComputerEnvironment) by Computer, SourceComputerId, UpdateID
    | summarize Computer=any(Computer), ComputerEnvironment=any(ComputerEnvironment),
missingCriticalUpdatesCount=countif(Classification has "Critical" and UpdateState=~"Needed" and
Approved!=false), missingSecurityUpdatesCount=countif(Classification has "Security" and
UpdateState=~"Needed" and Approved!=false), missingOtherUpdatesCount=countif(Classification !has "Critical"
and Classification !has "Security" and UpdateState=~"Needed" and Optional==false and Approved!=false),
lastAssessedTime=max(TimeGenerated), lastUpdateAgentSeenTime="" by SourceComputerId
    | extend compliance=iff(missingCriticalUpdatesCount > 0 or missingSecurityUpdatesCount > 0, 2, 1)
    | extend ComplianceOrder=iff(missingCriticalUpdatesCount > 0 or missingSecurityUpdatesCount > 0 or
missingOtherUpdatesCount > 0, 1, 3)
)
on SourceComputerId
| project id=SourceComputerId, displayName=Computer, sourceComputerId=SourceComputerId,
scopedToUpdatesSolution=true, missingCriticalUpdatesCount=coalesce(missingCriticalUpdatesCount, -1),
missingSecurityUpdatesCount=coalesce(missingSecurityUpdatesCount, -1),
missingOtherUpdatesCount=coalesce(missingOtherUpdatesCount, -1), compliance=coalesce(compliance, 4),
lastAssessedTime, lastUpdateAgentSeenTime, osType=2, environment=iff(ComputerEnvironment=~"Azure", 1, 2),
ComplianceOrder=coalesce(ComplianceOrder, 2) )
| order by ComplianceOrder asc, missingCriticalUpdatesCount desc, missingSecurityUpdatesCount desc,
missingOtherUpdatesCount desc, displayName asc
| project-away ComplianceOrder

```

Missing updates list

```

Update
| where TimeGenerated>ago(5h) and OSType=="Linux" and SourceComputerId in ((Heartbeat
| where TimeGenerated>ago(12h) and OSType=="Linux" and notempty(Computer)
| summarize arg_max(TimeGenerated, Solutions) by SourceComputerId
| where Solutions has "updates"
| distinct SourceComputerId))
| summarize hint.strategy=partitioned arg_max(TimeGenerated, UpdateState, Classification, BulletinUrl,
BulletinID) by SourceComputerId, Product, ProductArch
| where UpdateState=~"Needed"
| project-away UpdateState, TimeGenerated
| summarize computersCount=dcount(SourceComputerId, 2), ClassificationWeight=max(ifClassification has
"Critical", 4, iff(Classification has "Security", 2, 1))) by id=strcat(Product, "_", ProductArch),
displayName=Product, productArch=ProductArch, classification=Classification, InformationId=BulletinID,
InformationUrl=tostring(split(BulletinUrl, ";", 0)[0]), osType=1
| union(Update
| where TimeGenerated>ago(14h) and OSType!="Linux" and (Optional==false or Classification has "Critical" or
Classification has "Security") and SourceComputerId in ((Heartbeat
| where TimeGenerated>ago(12h) and OSType=~"Windows" and notempty(Computer)
| summarize arg_max(TimeGenerated, Solutions) by SourceComputerId
| where Solutions has "updates"
| distinct SourceComputerId))
| summarize hint.strategy=partitioned arg_max(TimeGenerated, UpdateState, Classification, Title, KBID,
PublishedDate, Approved) by Computer, SourceComputerId, UpdateID
| where UpdateState=~"Needed" and Approved!=false
| project-away UpdateState, Approved, TimeGenerated
| summarize computersCount=dcount(SourceComputerId, 2), displayName=any(Title),
publishedDate=min(PublishedDate), ClassificationWeight=max(ifClassification has "Critical", 4,
iff(Classification has "Security", 2, 1))) by id=strcat(UpdateID, "_", KBID), classification=Classification,
InformationId=strcat("KB", KBID), InformationUrl=iff(isnotempty(KBID),
strcat("https://support.microsoft.com/kb/", KBID), ""), osType=2
| sort by ClassificationWeight desc, computersCount desc, displayName asc
| extend informationLink=(iff(isnotempty(InformationId) and isnotempty(InformationUrl), toobject(strcat('{",
"uri": "", InformationUrl, '", "text": "", InformationId, '", "target": "blank" }')), toobject('')))
| project-away ClassificationWeight, InformationId, InformationUrl

```

Next steps

- For details of Azure Monitor logs, see [Azure Monitor logs](#).
- For help with alerts, see [Configure alerts](#).

Limit Update Management deployment scope

6/9/2021 • 2 minutes to read • [Edit Online](#)

This article describes how to work with scope configurations when using the [Update Management](#) feature to deploy updates and patches to your machines. For more information, see [Targeting monitoring solutions in Azure Monitor \(Preview\)](#).

About scope configurations

A scope configuration is a group of one or more saved searches (queries) used to limit the scope of Update Management to specific computers. The scope configuration is used within the Log Analytics workspace to target the computers to enable. When you add a computer to receive updates from Update Management, the computer is also added to a saved search in the workspace.

By default, Update Management creates a computer group named **Updates__MicrosoftDefaultComputerGroup** depending on how you enabled machines with Update Management:

- From the Automation account, you selected **+ Add Azure VMs**.
- From the Automation account, you selected **Manage machines**, and then you selected the option **Enable on all available machines** or you selected **Enable on selected machines**.

If one of the methods above is selected, this computer group is added to the **MicrosoftDefaultScopeConfig-Updates** scope configuration. You can also add one or more custom computer groups to this scope to match your management needs and control how specific computers are enabled for management with Update Management.

To remove one or more machines from the **Updates__MicrosoftDefaultComputerGroup** to stop managing them with Update Management, see [Remove VMs from Update Management](#).

Set the scope limit

To limit the scope for your Update Management deployment:

1. Sign in to the [Azure portal](#).
2. In the Azure portal, navigate to **Log Analytics workspaces**. Select your workspace from the list.
3. In your Log Analytics workspace, select **Scope Configurations (Preview)** from the left-hand menu.
4. Select the ellipsis to the right of the **MicrosoftDefaultScopeConfig-Updates** scope configuration, and select **Edit**.
5. In the editing pane, expand **Select Computer Groups**. The **Computer Groups** pane shows the saved searches that are added to the scope configuration. The saved search used by Update Management is:

NAME	CATEGORY	ALIAS
MicrosoftDefaultComputerGroup	Updates	Updates__MicrosoftDefaultComputerGroup

6. If you added a custom group, it is shown in the list. To deselect it, clear the checkbox to the left of the item. To add a custom group to the scope, select it and then when you are finished with your changes, click

Select.

7. On the **Edit scope configuration** page, click **OK** to save your changes.

Next steps

You can [query Azure Monitor logs](#) to analyze update assessments, deployments, and other related management tasks. It includes pre-defined queries to help you get started.

Remove Update Management from Automation account

11/2/2020 • 2 minutes to read • [Edit Online](#)

After you enable management of updates on your virtual machines using Azure Automation Update Management, you may decide to stop using it and remove the configuration from the account and linked Log Analytics workspace. This article tells you how to completely remove Update Management from the managed VMs, your Automation account, and Log Analytics workspace.

Sign into the Azure portal

Sign in to the [Azure portal](#).

Remove management of VMs

Before removing Update Management, you need to first stop managing your VMs. See [Remove VMs from Update Management](#) to unenroll them from the feature.

Remove UpdateManagement solution

Before you can unlink the Automation account from the workspace, you need to follow these steps to completely remove Update Management. You'll remove the **Updates** solution from the workspace.

1. Sign in to the [Azure portal](#).
2. In the Azure portal, select **All services**. In the list of resources, type **Log Analytics**. As you begin typing, the list filters suggestions based on your input. Select **Log Analytics**.
3. In your list of Log Analytics workspaces, select the workspace you chose when you enabled Update Management.
4. On the left, select **Solutions**.
5. In the list of solutions, select **Updates(workspace name)**. On the **Overview** page for the solution, select **Delete**. When prompted to confirm, select **Yes**.

Unlink workspace from Automation account

1. In the Azure portal, select **Automation Accounts**.
2. Open your Automation account and select **Linked workspace** under **Related Resources** on the left.
3. On the **Unlink workspace** page, select **Unlink workspace** and respond to prompts.

A screenshot of the Azure portal showing the 'Automation Accounts' blade for an account named 'MAIC-AA-Pri'. On the left, there's a navigation menu with 'Search (Ctrl+ /)', 'Credentials', 'Connections', 'Certificates', and 'Variables'. Under 'Related Resources', 'Linked workspace' is selected, followed by 'Event grid' and 'Start/Stop VM'. Under 'Account Settings', there are links for 'Properties', 'Keys', and 'Pricing'. The main content area is titled 'MAIC-AA-Pri | Linked workspace' and shows that the account is linked to a Log Analytics workspace named 'maic-la'. It includes buttons for 'Go to workspace' and 'Unlink workspace'. A note states: 'This Automation account is linked to the following Log Analytics workspace: maic-la'. Below this, there's a section for 'Unlink workspace' with a list of items to remove: 'Update Management', 'Change Tracking', and 'Start/Stop VMs during off-hours'. A note says: 'After you remove these solutions you can click Unlink workspace above to complete the process.' Another section says: 'If you use the Update Management solution you optionally may want to remove some items': 'Update schedules' (with a link to 'View schedules') and 'Hybrid worker groups created for the solution' (with a link to 'Learn how to remove a hybrid worker group').

While it attempts to unlink the Log Analytics workspace, you can track the progress under **Notifications** from the menu.

Alternatively, you can unlink your Log Analytics workspace from your Automation account from within the workspace:

1. In the Azure portal, select **Log Analytics**.
2. From the workspace, select **Automation Account** under **Related Resources**.
3. On the Automation Account page, select **Unlink account**.

While it attempts to unlink the Automation account, you can track the progress under **Notifications** from the menu.

Cleanup Automation account

If Update Management was configured to support earlier versions of Azure SQL monitoring, setup of the feature might have created Automation assets that you should remove. For Update Management, you might want to remove the following items that are no longer needed:

- Update schedules - Each has a name that matches the update deployment you created.
- Hybrid worker groups created for Update Management - Each is named similarly to *machine1.contoso.com_9ceb8108-26c9-4051-b6b3-227600d715c8*.

Next steps

To re-enable this feature, see [Enable Update Management from an Automation account](#), [Enable Update Management by browsing the Azure portal](#), [Enable Update Management from a runbook](#), or [Enable Update Management from an Azure VM](#).

Remove VMs from Update Management

6/9/2021 • 2 minutes to read • [Edit Online](#)

When you're finished managing updates on your VMs in your environment, you can stop managing VMs with the [Update Management](#) feature. To stop managing them, you will edit the saved search query

`MicrosoftDefaultComputerGroup` in your Log Analytics workspace that is linked to your Automation account.

Sign into the Azure portal

Sign in to the [Azure portal](#).

To remove your VMs

1. In the Azure portal, launch **Cloud Shell** from the top navigation of the Azure portal. If you are unfamiliar with Azure Cloud Shell, see [Overview of Azure Cloud Shell](#).
2. Use the following command to identify the UUID of a machine that you want to remove from management.

```
az vm show -g MyResourceGroup -n MyVm -d
```
3. In the Azure portal, navigate to **Log Analytics workspaces**. Select your workspace from the list.
4. In your Log Analytics workspace, select **Computer Groups** from the left-hand menu.
5. From **Computer Groups** in the right-hand pane, the **Saved groups** tab is shown by default.
6. From the table, click the icon **Run query** to the right of the item **MicrosoftDefaultComputerGroup** with the **Legacy category** value **Updates**.
7. In the query editor, review the query and find the UUID for the VM. Remove the UUID for the VM and repeat the steps for any other VMs you want to remove.

NOTE

For added protection, before making edits be sure to make a copy of the query. Then you can restore it if a problem occurs.

If you want to start with the original query and re-add machines in support of a cleanup or maintenance activity, copy the following query:

```
Heartbeat  
| where Computer in~ ("") or VMUUID in~ ("")  
| distinct Computer
```

8. Save the saved search when you're finished editing it by selecting **Save > Save as function** from the top bar. When prompted, specify the following:
 - **Name:** Updates__MicrosoftDefaultComputerGroup
 - **Save as computer Group** is selected
 - **Legacy category:** Updates

NOTE

Machines are still shown after you have unenrolled them because we report on all machines assessed in the last 24 hours. After removing the machine, you need to wait 24 hours before they are no longer listed.

Next steps

To re-enable managing your virtual machine, see [Enable Update Management by browsing the Azure portal](#) or [Enable Update Management from an Azure VM](#).

Troubleshoot feature deployment issues

8/20/2021 • 6 minutes to read • [Edit Online](#)

You might receive error messages when you deploy the Azure Automation Update Management feature or the Change Tracking and Inventory feature on your VMs. This article describes the errors that might occur and how to resolve them.

Known issues

Scenario: Renaming a registered node requires unregister or register again

Issue

A node is registered to Azure Automation, and then the operating system computer name is changed. Reports from the node continue to appear with the original name.

Cause

Renaming registered nodes doesn't update the node name in Azure Automation.

Resolution

Unregister the node from Azure Automation State Configuration, and then register it again. Reports published to the service before that time will no longer be available.

Scenario: Re-signing certificates via HTTPS proxy isn't supported

Issue

When you connect through a proxy that terminates HTTPS traffic and then re-encrypts the traffic using a new certificate, the service doesn't allow the connection.

Cause

Azure Automation doesn't support re-signing certificates used to encrypt traffic.

Resolution

There's currently no workaround for this issue.

General errors

Scenario: Feature deployment fails with the message "The solution cannot be enabled"

Issue

You receive one of the following messages when you attempt to enable a feature on a VM:

The solution cannot be enabled due to missing permissions for the virtual machine or deployments

The solution cannot be enabled on this VM because the permission to read the workspace is missing

Cause

This error is caused by incorrect or missing permissions on the VM or workspace, or for the user.

Resolution

Ensure that you have correct [feature deployment permissions](#), and then try to deploy the feature again. If you receive the error message

The solution cannot be enabled on this VM because the permission to read the workspace is missing, see the following [troubleshooting information](#).

Scenario: Feature deployment fails with the message "Failed to configure automation account for diagnostic logging"

Issue

You receive the following message when you attempt to enable a feature on a VM:

```
Failed to configure automation account for diagnostic logging
```

Cause

This error can be caused if the pricing tier doesn't match the subscription's billing model. For more information, see [Monitoring usage and estimated costs in Azure Monitor](#).

Resolution

Create your Log Analytics workspace manually, and repeat the feature deployment process to select the workspace created.

Scenario: ComputerGroupQueryFormatError

Issue

This error code means that the saved search computer group query used to target the feature isn't formatted correctly.

Cause

You might have altered the query, or the system might have altered it.

Resolution

You can delete the query for the feature and then enable the feature again, which re-creates the query. The query can be found in your workspace under [Saved searches](#). The name of the query is **MicrosoftDefaultComputerGroup**, and the category of the query is the name of the associated feature. If multiple features are enabled, the **MicrosoftDefaultComputerGroup** query shows multiple times under [Saved searches](#).

Scenario: PolicyViolation

Issue

This error code indicates that the deployment failed due to violation of one or more Azure Policy assignments.

Cause

An Azure Policy assignment is blocking the operation from completing.

Resolution

To successfully deploy the feature, you must consider altering the indicated policy definition. Because there are many different types of policy definitions that can be defined, the changes required depend on the policy definition that's violated. For example, if a policy definition is assigned to a resource group that denies permission to change the contents of some contained resources, you might choose one of these fixes:

- Remove the policy assignment altogether.
- Try to enable the feature for a different resource group.
- Retarget the policy assignment to a specific resource, for example, an Automation account.
- Revise the set of resources that the policy definition is configured to deny.

Check the notifications in the upper-right corner of the Azure portal, or go to the resource group that contains your Automation account and select **Deployments** under **Settings** to view the failed deployment. To learn more about Azure Policy, see [Overview of Azure Policy](#).

Scenario: Errors trying to unlink a workspace

Issue

You receive the following error message when you try to unlink a workspace:

The link cannot be updated or deleted because it is linked to Update Management and/or ChangeTracking Solutions.

Cause

This error occurs when you still have features active in your Log Analytics workspace that depend on your Automation account and Log Analytics workspace being linked.

Resolution

Remove the resources for the following features from your workspace if you're using them:

- Update Management
- Change Tracking and Inventory
- Start/Stop VMs during off-hours

After you remove the feature resources, you can unlink your workspace. It's important to clean up any existing artifacts from these features from your workspace and your Automation account:

- For Update Management, remove **Update Deployments (Schedules)** from your Automation account.
- For Start/Stop VMs during off-hours, remove any locks on feature components in your Automation account under **Settings > Locks**. For more information, see [Remove the feature](#).

Log Analytics for Windows extension failures

NOTE

As part of the ongoing transition from Microsoft Operations Management Suite to Azure Monitor, the Operations Management Suite Agent for Windows or Linux will be referred to as the Log Analytics agent for Windows and Log Analytics agent for Linux.

An installation of the Log Analytics agent for Windows extension can fail for a variety of reasons. The following section describes feature deployment issues that can cause failures during deployment of the Log Analytics agent for Windows extension.

NOTE

Log Analytics agent for Windows is the name used currently in Azure Automation for the Microsoft Monitoring Agent (MMA).

Scenario: An exception occurred during a WebClient request

The Log Analytics for Windows extension on the VM is unable to communicate with external resources and the deployment fails.

Issue

The following are examples of error messages that are returned:

Please verify the VM has a running VM agent, and can establish outbound connections to Azure storage.

'Manifest download error from
https://<endpoint>/<endpointId>/Microsoft.EnterpriseCloud.Monitoring_MicrosoftMonitoringAgent_australiaeast_manifest.xml. Error: UnknownError. An exception occurred during a WebClient request.'

Cause

Some potential causes of this error are:

- A proxy configured in the VM only allows specific ports.
- A firewall setting has blocked access to the required ports and addresses.

Resolution

Ensure that you have the proper ports and addresses open for communication. For a list of ports and addresses, see [Planning your network](#).

Scenario: Install failed because of transient environment issues

The installation of the Log Analytics for Windows extension failed during deployment because of another installation or action blocking the installation.

Issue

The following are examples of error messages that might be returned:

```
The Microsoft Monitoring Agent failed to install on this machine. Please try to uninstall and reinstall the extension. If the issue persists, please contact support.
```

```
'Install failed for plugin (name: Microsoft.EnterpriseCloud.Monitoring.MicrosoftMonitoringAgent, version 1.0.11081.4) with exception Command C:\Packages\Plugins\Microsoft.EnterpriseCloud.Monitoring.MicrosoftMonitoringAgent\1.0.11081.4\MMAExtensionInstall.exe of Microsoft.EnterpriseCloud.Monitoring.MicrosoftMonitoringAgent has exited with Exit code: 1618'
```

```
'Install failed for plugin (name: Microsoft.EnterpriseCloud.Monitoring.MicrosoftMonitoringAgent, version 1.0.11081.2) with exception Command C:\Packages\Plugins\Microsoft.EnterpriseCloud.Monitoring.MicrosoftMonitoringAgent\1.0.11081.2\MMAExtensionInstall.exe of Microsoft.EnterpriseCloud.Monitoring.MicrosoftMonitoringAgent has exited with Exit code: 1601'
```

Cause

Some potential causes of this error are:

- Another installation is in progress.
- The system is triggered to reboot during template deployment.

Resolution

This error is transient in nature. Retry the deployment to install the extension.

Scenario: Installation timeout

The installation of the Log Analytics for Windows extension didn't complete because of a timeout.

Issue

The following is an example of an error message that might be returned:

```
Install failed for plugin (name: Microsoft.EnterpriseCloud.Monitoring.MicrosoftMonitoringAgent, version 1.0.11081.4) with exception Command C:\Packages\Plugins\Microsoft.EnterpriseCloud.Monitoring.MicrosoftMonitoringAgent\1.0.11081.4\MMAExtensionInstall.exe of Microsoft.EnterpriseCloud.Monitoring.MicrosoftMonitoringAgent has exited with Exit code: 15614
```

Cause

This type of error occurs because the VM is under a heavy load during installation.

Resolution

Try to install the Log Analytics agent for Windows extension when the VM is under a lower load.

Next steps

If you don't see your problem here or you can't resolve your issue, try one of the following channels for additional support:

- Get answers from Azure experts through [Azure Forums](#).
- Connect with [@AzureSupport](#), the official Microsoft Azure account for improving customer experience. Azure Support connects the Azure community to answers, support, and experts.
- File an Azure support incident. Go to the [Azure support site](#), and select **Get Support**.

Troubleshoot Update Management issues

8/24/2021 • 26 minutes to read • [Edit Online](#)

This article discusses issues that you might run into when using the Update Management feature to assess and manage updates on your machines. There's an agent troubleshooter for the Hybrid Runbook Worker agent to help determine the underlying problem. To learn more about the troubleshooter, see [Troubleshoot Windows update agent issues](#) and [Troubleshoot Linux update agent issues](#). For other feature deployment issues, see [Troubleshoot feature deployment issues](#).

NOTE

If you run into problems when deploying Update Management on a Windows machine, open the Windows Event Viewer, and check the **Operations Manager** event log under **Application and Services Logs** on the local machine. Look for events with event ID 4502 and event details that contain

```
Microsoft.EnterpriseManagement.HealthService.AzureAutomation.HybridAgent .
```

Scenario: Windows Defender update always show as missing

Issue

Definition update for Windows Defender ([KB2267602](#)) always shows as missing in an assessment when it's installed and shows as up to date when verified from Windows Update history.

Cause

Definition updates are published multiple times in a single day. As a result, you could see multiple releases of KB2267602 published in a single day, but with a different update ID and version.

Update Management assessment runs once in 11 hours. In this example, at 10:00 AM an assessment ran and version 1.237.316.0 was available at the time. When you search the **Update** table in your Log Analytics workspace, the Definition update 1.237.316.0 is shown with an **UpdateState** of **Needed**. If a scheduled deployment runs a few hours later, let's say 1:00 PM and version 1.237.316.0 is still available or a newer version is, the newer version is installed and this is reflected in the record written to the **UpdateRunProgress** table. However, in the **Update** table, it would still show version 1.237.316.0 as **Needed** until the next assessment is run. When the assessment runs again, there may not be a newer definition update available, so the **Update** table would not show the definition update version 1.237.316.0 as missing or a newer version available as needed. Because of the frequency of definition updates, there could be multiple versions returned in the log search.

Resolution

Run the following log query to confirm definition updates installed are being properly reported. This query returns the time generated, version, and update ID of KB2267602 in the **Updates** table. Replace the value for *Computer* with the fully qualified name of the machine.

```
Update
| where TimeGenerated > ago(14h) and OSType != "Linux" and (Optional == false or Classification has "Critical" or Classification has "Security") and SourceComputerId in (
    Heartbeat
    | where TimeGenerated > ago(12h) and OSType =~ "Windows" and notempty(Computer)
    | summarize arg_max(TimeGenerated, Solutions) by SourceComputerId
    | where Solutions has "updates"
    | distinct SourceComputerId)
| summarize hint.strategy=partitioned arg_max(TimeGenerated, *) by Computer, SourceComputerId, UpdateID
| where UpdateState =~ "Needed" and Approved != false and Computer == "<computerName>"
| render table
```

Your query results should return something similar to the following:

TimeGenerated [UTC]	Computer	SourceComputerId	UpdateID	Title
12/3/2020, 10:24:51.637 AM	win-test-hw	82e3c8a5-b1ec-4db5-9104-d04002d4f451	53d7a2a7-e5f2-411b-b155-2c31f59e783a	Security Intelligence Update for Microsoft Defender Antivirus - KB2267602 Version (1.327.1991.0)
...				
Computer	win-test-hw			
SourceComputerId		82e3c8a5-b1ec-4db5-9104-d04002d4f451		
UpdateID			53d7a2a7-e5f2-411b-b155-2c31f59e783a	
TimeGenerated [UTC]				2020-12-03T10:24:51Z
TenantId				0314fa67-b7e3-40ef-8dec-1d9f5b7a6632
SourceSystem				OpsManager
MG				00000000-0000-0000-0000-000000000001
ManagementGroupName				AOI-0314fa67-b7e3-40ef-8dec-1d9f5b7a6632
Title				Security Intelligence Update for Microsoft Defender Antivirus - KB2267602 Version (1.327.1991.0)
Classification				Definition Updates

Run the following log query to get the time generated, version, and update ID of KB2267602 in the **UpdatesRunProgress** table. This query helps us understand if it was installed from Update Management or if it was auto-installed on the machine from Microsoft Update. You need to replace the value for *CorrelationId* with the runbook job GUID (that is, the **MasterJOBID** property value from the **Patch-MicrosoftOMSComputer** runbook job) for the update, and *SourceComputerId* with the GUID of the machine.

```
UpdateRunProgress
| where OSType!="Linux" and CorrelationId=="<master job id>" and SourceComputerId=="<source computer id>"
| summarize arg_max(TimeGenerated, Title, InstallationStatus) by UpdateID
| project TimeGenerated, id=UpdateID, displayName=Title, InstallationStatus
```

Your query results should return something similar to the following:

TimeGenerated [UTC]	id	displayName	InstallationStatus
12/3/2020, 8:34:35.107 PM	40443be5-0283-4b05-8bdb-9604b7f5d999	Security Intelligence Update for Microsoft Defender Antivirus - KB2267602 (Version 1.327.2014.0)	Succeeded
TimeGenerated [UTC] 2020-12-03T20:34:35:107Z			
id	40443be5-0283-4b05-8bdb-9604b7f5d999		
displayName		Security Intelligence Update for Microsoft Defender Antivirus - KB2267602 (Version 1.327.2014.0)	
InstallationStatus			Succeeded

If the **TimeGenerated** value for the log query results from the **Updates** table is earlier than the timestamp (that is, value of **TimeGenerated**) of the update installation on machine or from the log query results from the **UpdateRunProgress** table, then wait for the next assessment. Afterwards, run the log query against the **Updates** table again. Either an update for KB2267602 won't appear or it appears with a newer version. However, even after the most recent assessment if same version shows up as **Needed** in the **Updates** table but it is already installed, you should open an Azure support incident.

Scenario: Linux updates shown as pending and those installed vary

Issue

For your Linux machine, Update Management shows specific updates available under classification **Security** and **Others**. But when an update schedule is run on the machine, for example to install only updates matching the **Security** classification, the updates installed are different from or a subset of the updates shown earlier matching that classification.

Cause

When an assessment of OS updates pending for your Linux machine is done, [Open Vulnerability and Assessment Language](#) (OVAL) files provided by the Linux distro vendor is used by Update Management for classification. Categorization is done for Linux updates as **Security** or **Others**, based on the OVAL files which states updates addressing security issues or vulnerabilities. But when the update schedule is run, it executes on the Linux machine using the appropriate package manager like YUM, APT or ZYPPER to install them. The package manager for the Linux distro may have a different mechanism to classify updates, where the results may differ from the ones obtained from OVAL files by Update Management.

Resolution

You can manually check the Linux machine, the applicable updates, and their classification per the distro's package manager. To understand which updates are classified as **Security** by your package manager, run the following commands.

For YUM, the following command returns a non-zero list of updates categorized as **Security** by Red Hat. Note that in the case of CentOS, it always returns an empty list and no security classification occurs.

```
sudo yum -q --security check-update
```

For ZYPPER, the following command returns a non-zero list of updates categorized as **Security** by SUSE.

```
sudo LANG=en_US.UTF8 zypper --non-interactive patch --category security --dry-run
```

For APT, the following command returns a non-zero list of updates categorized as **Security** by Canonical for Ubuntu Linux distros.

```
sudo grep security /etc/apt/sources.list > /tmp/oms-update-security.list LANG=en_US.UTF8 sudo apt-get -s dist-upgrade -oDir::Etc::Sourcelist=/tmp/oms-update-security.list
```

From this list you then run the command `grep ^Inst` to get all the pending security updates.

Scenario: You receive the error "Failed to enable the Update solution"

Issue

When you try to enable Update Management in your Automation account, you get the following error:

```
Error details: Failed to enable the Update solution
```

Cause

This error can occur for the following reasons:

- The network firewall requirements for the Log Analytics agent might not be configured correctly. This situation can cause the agent to fail when resolving the DNS URLs.
- Update Management targeting is misconfigured and the machine isn't receiving updates as expected.
- You might also notice that the machine shows a status of `Non-compliant` under **Compliance**. At the same time, **Agent Desktop Analytics** reports the agent as `Disconnected`.

Resolution

- Run the troubleshooter for [Windows](#) or [Linux](#), depending on the OS.

- Go to [Network configuration](#) to learn about which addresses and ports must be allowed for Update Management to work.
- Check for scope configuration problems. [Scope configuration](#) determines which machines are configured for Update Management. If your machine is showing up in your workspace but not in Update Management, you must set the scope configuration to target the machines. To learn about the scope configuration, see [Enable machines in the workspace](#).
- Remove the worker configuration by following the steps in [Remove the Hybrid Runbook Worker from an on-premises Windows computer](#) or [Remove the Hybrid Runbook Worker from an on-premises Linux computer](#).

Scenario: Superseded update indicated as missing in Update Management

Issue

Old updates are appearing for an Automation account as missing even though they've been superseded. A superseded update is one that you don't have to install because a later update that corrects the same vulnerability is available. Update Management ignores the superseded update and makes it not applicable in favor of the superseding update. For information about a related issue, see [Update is superseded](#).

Cause

Superseded updates aren't declined in Windows Server Update Services (WSUS) so that they can be considered not applicable.

Resolution

When a superseded update becomes 100 percent not applicable, you should change the approval state of that update to [Declined](#) in WSUS. To change approval state for all your updates:

1. In the Automation account, select **Update Management** to view machine status. See [View update assessments](#).
2. Check the superseded update to make sure that it's 100 percent not applicable.
3. On the WSUS server the machines report to, [decline the update](#).
4. Select **Computers** and, in the **Compliance** column, force a rescan for compliance. See [Manage updates for VMs](#).
5. Repeat the steps above for other superseded updates.
6. For Windows Server Update Services (WSUS), clean all superseded updates to refresh the infrastructure using the WSUS [Server cleanup Wizard](#).
7. Repeat this procedure regularly to correct the display issue and minimize the amount of disk space used for update management.

Scenario: Machines don't show up in the portal under Update Management

Issue

Your machines have the following symptoms:

- Your machine shows [Not configured](#) from the Update Management view of a VM.
- Your machines are missing from the Update Management view of your Azure Automation account.
- You have machines that show as [Not assessed](#) under **Compliance**. However, you see heartbeat data in Azure Monitor logs for the Hybrid Runbook Worker but not for Update Management.

Cause

This issue can be caused by local configuration issues or by improperly configured scope configuration. Possible specific causes are:

- You might have to re-register and reinstall the Hybrid Runbook Worker.
- You might have defined a quota in your workspace that's been reached and that's preventing further data storage.

Resolution

1. Run the troubleshooter for [Windows](#) or [Linux](#), depending on the OS.
2. Make sure that your machine is reporting to the correct workspace. For guidance on how to verify this aspect, see [Verify agent connectivity to Azure Monitor](#). Also make sure that this workspace is linked to your Azure Automation account. To confirm, go to your Automation account and select **Linked workspace** under **Related Resources**.
3. Make sure that the machines show up in the Log Analytics workspace linked to your Automation account. Run the following query in the Log Analytics workspace.

```
Heartbeat  
| summarize by Computer, Solutions
```

If you don't see your machine in the query results, it hasn't recently checked in. There's probably a local configuration issue and you should [reinstall the agent](#).

If your machine is listed in the query results, verify under the **Solutions** property that **updates** is listed. This verifies it is registered with Update Management. If it is not, check for scope configuration problems. The [scope configuration](#) determines which machines are configured for Update Management. To configure the scope configuration for the target the machine, see [Enable machines in the workspace](#).

4. In your workspace, run this query.

```
Operation  
| where OperationCategory == 'Data Collection Status'  
| sort by TimeGenerated desc
```

If you get a

`Data collection stopped due to daily limit of free data reached. Ingestion status = OverQuota` result, the quota defined on your workspace has been reached, which has stopped data from being saved. In your workspace, go to **data volume management** under **Usage and estimated costs**, and change or remove the quota.

5. If your issue is still unresolved, follow the steps in [Deploy a Windows Hybrid Runbook Worker](#) to reinstall the Hybrid Worker for Windows. For Linux, follow the steps in [Deploy a Linux Hybrid Runbook Worker](#).

Scenario: Unable to register Automation resource provider for subscriptions

Issue

When you work with feature deployments in your Automation account, the following error occurs:

```
Error details: Unable to register Automation Resource Provider for subscriptions
```

Cause

The Automation resource provider isn't registered in the subscription.

Resolution

To register the Automation resource provider, follow these steps in the Azure portal.

1. In the Azure service list at the bottom of the portal, select **All services**, and then select **Subscriptions** in the General service group.
2. Select your subscription.
3. Under **Settings**, select **Resource Providers**.
4. From the list of resource providers, verify that the Microsoft.Automation resource provider is registered.
5. If it's not listed, register the Microsoft.Automation provider by following the steps at [Resolve errors for resource provider registration](#).

Scenario: Scheduled update did not patch some machines

Issue

Machines included in an update preview don't all appear in the list of machines patched during a scheduled run, or VMs for selected scopes of a dynamic group are not showing up in the update preview list in the portal.

The update preview list consists of all machines retrieved by an [Azure Resource Graph](#) query for the selected scopes. The scopes are filtered for machines that have a system Hybrid Runbook Worker installed and for which you have access permissions.

Cause

This issue can have one of the following causes:

- The subscriptions defined in the scope in a dynamic query aren't configured for the registered Automation resource provider.
- The machines weren't available or didn't have appropriate tags when the schedule executed.
- You don't have the correct access on the selected scopes.
- The Azure Resource Graph query doesn't retrieve the expected machines.
- The system Hybrid Runbook Worker isn't installed on the machines.

Resolution

Subscriptions not configured for registered Automation resource provider

If your subscription isn't configured for the Automation resource provider, you can't query or fetch information on machines in that subscription. Use the following steps to verify the registration for the subscription.

1. In the [Azure portal](#), access the Azure service list.
2. Select **All services**, and then select **Subscriptions** in the General service group.
3. Find the subscription defined in the scope for your deployment.
4. Under **Settings**, choose **Resource Providers**.
5. Verify that the Microsoft.Automation resource provider is registered.
6. If it's not listed, register the Microsoft.Automation provider by following the steps at [Resolve errors for resource provider registration](#).

Machines not available or not tagged correctly when schedule executed

Use the following procedure if your subscription is configured for the Automation resource provider, but running the update schedule with the specified [dynamic groups](#) missed some machines.

1. In the Azure portal, open the Automation account and select **Update Management**.
2. Check [Update Management history](#) to determine the exact time when the update deployment was run.
3. For machines that you suspect to have been missed by Update Management, use Azure Resource Graph

(ARG) to [locate machine changes](#).

4. Search for changes over a considerable period, such as one day, before the update deployment was run.
5. Check the search results for any systemic changes, such as delete or update changes, to the machines in this period. These changes can alter machine status or tags so that machines aren't selected in the machine list when updates are deployed.
6. Adjust the machines and resource settings as necessary to correct for machine status or tag issues.
7. Rerun the update schedule to ensure that deployment with the specified dynamic groups includes all machines.

Incorrect access on selected scopes

The Azure portal only displays machines for which you have write access in a given scope. If you don't have the correct access for a scope, see [Tutorial: Grant a user access to Azure resources using the Azure portal](#).

Resource Graph query doesn't return expected machines

Follow the steps below to find out if your queries are working correctly.

1. Run an Azure Resource Graph query formatted as shown below in the Resource Graph explorer blade in Azure portal. If you are new to Azure Resource Graph, see this [quickstart](#) to learn how to work with Resource Graph explorer. This query mimics the filters you selected when you created the dynamic group in Update Management. See [Use dynamic groups with Update Management](#).

```
where (subscriptionId in~ ("<subscriptionId1>", "<subscriptionId2>") and type =~ "microsoft.compute/virtualmachines" and properties.storageProfile.osDisk.osType == "<Windows/Linux>" and resourceGroup in~ ("<resourceGroupName1>","<resourceGroupName2>") and location in~ ("<location1>","<location2>") )  
| project id, location, name, tags = todynamic(tolower(tostring(tags)))  
| where (tags[tolower("<tagKey1>")] =~ "<tagValue1>" and tags[tolower("<tagKey2>")] =~ "<tagValue2>") // use this if "All" option selected for tags  
| where (tags[tolower("<tagKey1>")] =~ "<tagValue1>" or tags[tolower("<tagKey2>")] =~ "<tagValue2>") // use this if "Any" option selected for tags  
| project id, location, name, tags
```

Here is an example:

```
where (subscriptionId in~ ("20780d0a-b422-4213-979b-6c919c91ace1", "af52d412-a347-4bc6-8cb7-4780fbb00490") and type =~ "microsoft.compute/virtualmachines" and properties.storageProfile.osDisk.osType == "Windows" and resourceGroup in~ ("testRG","withinvnet-2020-01-06-10-global-resources-southindia") and location in~ ("australiacentral","australiacentral2","brazilsouth") )  
| project id, location, name, tags = todynamic(tolower(tostring(tags)))  
| where (tags[tolower("ms-resource-usage")] =~ "azure-cloud-shell" and tags[tolower("temp")] =~ "temp")  
| project id, location, name, tags
```

2. Check to see if the machines you're looking for are listed in the query results.
3. If the machines aren't listed, there is probably an issue with the filter selected in the dynamic group. Adjust the group configuration as needed.

Hybrid Runbook Worker not installed on machines

Machines do appear in Azure Resource Graph query results, but still don't show up in the dynamic group preview. In this case, the machines might not be designated as system Hybrid Runbook workers and thus can't run Azure Automation and Update Management jobs. To ensure that the machines you're expecting to see are set up as system Hybrid Runbook Workers:

1. In the Azure portal, go to the Automation account for a machine that is not appearing correctly.
2. Select **Hybrid worker groups** under **Process Automation**.
3. Select the **System hybrid worker groups** tab.

4. Validate that the hybrid worker is present for that machine.
5. If the machine is not set up as a system Hybrid Runbook Worker, review the methods to enable using one of the following methods:
 - From your [Automation account](#) for one or more Azure and non-Azure machines, including Arc-enabled servers.
 - Using the [Enable-AutomationSolution runbook](#) to automate onboarding Azure VMs.
 - For a [selected Azure VM](#) from the [Virtual machines](#) page in the Azure portal. This scenario is available for Linux and Windows VMs.
 - For [multiple Azure VMs](#) by selecting them from the [Virtual machines](#) page in the Azure portal.

The method to enable is based on the environment the machine is running in.

6. Repeat the steps above for all machines that have not been displaying in the preview.

Scenario: Update Management components enabled, while VM continues to show as being configured

Issue

You continue to see the following message on a VM 15 minutes after deployment begins:

The components for the 'Update Management' solution have been enabled, and now this virtual machine is being configured. Please be patient, as this can sometimes take up to 15 minutes.

Cause

This error can occur for the following reasons:

- Communication with the Automation account is being blocked.
- There is a duplicate computer name with different source computer IDs. This scenario occurs when a VM with a particular computer name is created in different resource groups and is reporting to the same Log Analytics workspace in the subscription.
- The VM image being deployed might come from a cloned machine that wasn't prepared with System Preparation (sysprep) with the Log Analytics agent for Windows installed.

Resolution

To help in determining the exact problem with the VM, run the following query in the Log Analytics workspace that's linked to your Automation account.

```
Update
| where Computer contains "fillInMachineName"
| project TimeGenerated, Computer, SourceComputerId, Title, UpdateState
```

Communication with Automation account blocked

Go to [Network planning](#) to learn about which addresses and ports must be allowed for Update Management to work.

Duplicate computer name

Rename your VMs to ensure unique names in their environment.

Deployed image from cloned machine

If you're using a cloned image, different computer names have the same source computer ID. In this case:

1. In your Log Analytics workspace, remove the VM from the saved search for the [MicrosoftDefaultScopeConfig-Updates](#) scope configuration if it's shown. Saved searches can be found under **General** in your workspace.

2. Run the following cmdlet.

```
Remove-Item -Path "HKLM:\software\microsoft\hybridrunbookworker" -Recurse -Force
```

3. Run `Restart-Service HealthService` to restart the health service. This operation recreates the key and generates a new UUID.
4. If this approach doesn't work, run sysprep on the image first and then install the Log Analytics agent for Windows.

Scenario: You receive a linked subscription error when you create an update deployment for machines in another Azure tenant

Issue

You encounter the following error when you try to create an update deployment for machines in another Azure tenant:

```
The client has permission to perform action 'Microsoft.Compute/virtualMachines/write' on scope '/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/resourceGroupName/providers/Microsoft.Automation/automationAccounts/automationAccountName/softwareUpdateConfigurations/updateDeploymentName', however the current tenant '00000000-0000-0000-0000-000000000000' is not authorized to access linked subscription '00000000-0000-0000-0000-000000000000'.
```

Cause

This error occurs when you create an update deployment that has Azure VMs in another tenant that's included in an update deployment.

Resolution

Use the following workaround to get these items scheduled. You can use the [New-AzAutomationSchedule](#) cmdlet with the `ForUpdateConfiguration` parameter to create a schedule. Then, use the [New-AzAutomationSoftwareUpdateConfiguration](#) cmdlet and pass the machines in the other tenant to the `NonAzureComputer` parameter. The following example shows how to do this:

```
$nonAzurecomputers = @("server-01", "server-02")

$startTime = ([DateTime]::Now).AddMinutes(10)

$s = New-AzAutomationSchedule -ResourceGroupName mygroup -AutomationAccountName myaccount -Name myupdateconfig -Description test-OneTime -OneTime -StartTime $startTime -ForUpdateConfiguration

New-AzAutomationSoftwareUpdateConfiguration -ResourceGroupName $rg -AutomationAccountName $aa -Schedule $s -Windows -AzureVMResourceId $azureVMIdsW -NonAzureComputer $nonAzurecomputers -Duration (New-Timespan -Hours 2) -IncludedUpdateClassification Security,UpdateRollup -ExcludedKbNumber KB01,KB02 -IncludedKbNumber KB100
```

Scenario: Unexplained reboots

Issue

Even though you've set the **Reboot Control** option to **Never Reboot**, machines are still rebooting after updates are installed.

Cause

Windows Update can be modified by several registry keys, any of which can modify reboot behavior.

Resolution

Review the registry keys listed under [Configuring Automatic Updates by editing the registry](#) and [Registry keys used to manage restart](#) to make sure your machines are configured properly.

Scenario: Machine shows "Failed to start" in an update deployment

Issue

A machine shows a `Failed to start` or `Failed` status. When you view the specific details for the machine, you see the following error:

For one or more machines in schedule, UM job run resulted in either Failed or Failed to start state. Guide available at <https://aka.ms/UMSucrFailed>.

Cause

This error can occur for one of the following reasons:

- The machine doesn't exist anymore.
- The machine is turned off and unreachable.
- The machine has a network connectivity issue, and therefore the hybrid worker on the machine is unreachable.
- There was an update to the Log Analytics agent that changed the source computer ID.
- Your update run was throttled if you hit the limit of 200 concurrent jobs in an Automation account. Each deployment is considered a job, and each machine in an update deployment counts as a job. Any other automation job or update deployment currently running in your Automation account counts toward the concurrent job limit.

Resolution

You can retrieve more details programmatically by using the REST API. See [Software Update Configuration Machine Runs](#) for information on retrieving either a list of update configuration machine runs, or a single software update configuration machine run by ID.

When applicable, use [dynamic groups](#) for your update deployments. In addition, you can take the following steps.

1. Verify that your machine or server meets the [requirements](#).
2. Verify connectivity to the Hybrid Runbook Worker using the Hybrid Runbook Worker agent troubleshooter.
To learn more about the troubleshooter, see [Troubleshoot update agent issues](#).

Scenario: Updates are installed without a deployment

Issue

When you enroll a Windows machine in Update Management, you see updates installed without a deployment.

Cause

On Windows, updates are installed automatically as soon as they're available. This behavior can cause confusion if you didn't schedule an update to be deployed to the machine.

Resolution

The `HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\WindowsUpdate\AU` registry key defaults to a setting of 4: `auto download and install`.

For Update Management clients, we recommend setting this key to 3: `auto download but do not auto install`.

For more information, see [Configuring Automatic Updates](#).

Scenario: Machine is already registered to a different account

Issue

You receive the following error message:

```
Unable to Register Machine for Patch Management, Registration Failed with Exception  
System.InvalidOperationException: {"Message":"Machine is already registered to a different account."}
```

Cause

The machine has already been deployed to another workspace for Update Management.

Resolution

1. Follow the steps under [Machines don't show up in the portal under Update Management](#) to make sure the machine is reporting to the correct workspace.
2. Clean up artifacts on the machine by [deleting the hybrid runbook group](#), and then try again.

Scenario: Machine can't communicate with the service

Issue

You receive one of the following error messages:

```
Unable to Register Machine for Patch Management, Registration Failed with Exception  
System.Net.Http.HttpRequestException: An error occurred while sending the request. --->  
System.Net.WebException: The underlying connection was closed: An unexpected error occurred on a receive. -->  
-> System.ComponentModel.Win32Exception: The client and server can't communicate, because they do not  
possess a common algorithm
```

```
Unable to Register Machine for Patch Management, Registration Failed with Exception  
Newtonsoft.Json.JsonReaderException: Error parsing positive infinity value.
```

```
The certificate presented by the service <wsid>.oms.opinsights.azure.com was not issued by a certificate  
authority used for Microsoft services. Contact your network administrator to see if they are running a proxy  
that intercepts TLS/SSL communication.
```

```
Access is denied. (Exception from HRESULT: 0x80070005(E_ACCESSDENIED))
```

Cause

A proxy, gateway, or firewall might be blocking network communication.

Resolution

Review your networking and make sure appropriate ports and addresses are allowed. See [network requirements](#) for a list of ports and addresses that are required by Update Management and Hybrid Runbook Workers.

Scenario: Unable to create self-signed certificate

Issue

You receive one of the following error messages:

```
Unable to Register Machine for Patch Management, Registration Failed with Exception  
AgentService.HybridRegistration. PowerShell.Certificates.CertificateCreationException: Failed to create a  
self-signed certificate. ---> System.UnauthorizedAccessException: Access is denied.
```

Cause

The Hybrid Runbook Worker couldn't generate a self-signed certificate.

Resolution

Verify that the system account has read access to the C:\ProgramData\Microsoft\Crypto\RSA folder, and try

again.

Scenario: The scheduled update failed with a MaintenanceWindowExceeded error

Issue

The default maintenance window for updates is 120 minutes. You can increase the maintenance window to a maximum of 6 hours, or 360 minutes. You might receive the error message

For one or more machines in schedule, UM job run resulted in Maintenance Window Exceeded state. Guide available at <https://aka.ms/UMSucrMwExceeded>.

Resolution

To understand why this occurred during an update run after it starts successfully, [check the job output](#) from the affected machine in the run. You might find specific error messages from your machines that you can research and take action on.

You can retrieve more details programmatically by using the REST API. See [Software Update Configuration Machine Runs](#) for information on retrieving either a list of update configuration machine runs, or a single software update configuration machine run by ID.

Edit any failing scheduled update deployments, and increase the maintenance window.

For more information on maintenance windows, see [Install updates](#).

Scenario: Machine shows as "Not assessed" and shows an HRESULT exception

Issue

- You have machines that show as **Not assessed** under **Compliance**, and you see an exception message below them.
- You see an HRESULT error code in the portal.

Cause

The Update Agent (Windows Update Agent on Windows; the package manager for a Linux distribution) isn't configured correctly. Update Management relies on the machine's Update Agent to provide the updates that are needed, the status of the patch, and the results of deployed patches. Without this information, Update Management can't properly report on the patches that are needed or installed.

Resolution

Try to perform updates locally on the machine. If this operation fails, it typically means that there's an update agent configuration error.

This problem is frequently caused by network configuration and firewall issues. Use the following checks to correct the issue.

- For Linux, check the appropriate documentation to make sure you can reach the network endpoint of your package repository.
- For Windows, check your agent configuration as listed in [Updates aren't downloading from the intranet endpoint \(WSUS/SCCM\)](#).
 - If the machines are configured for Windows Update, make sure that you can reach the endpoints described in [Issues related to HTTP/proxy](#).
 - If the machines are configured for Windows Server Update Services (WSUS), make sure that you can reach the WSUS server configured by the [WUServer registry key](#).

If you see an HRESULT, double-click the exception displayed in red to see the entire exception message. Review the following table for potential resolutions or recommended actions.

EXCEPTION	RESOLUTION OR ACTION
Exception from HRESULT: 0x.....C	Search the relevant error code in Windows update error code list to find additional details about the cause of the exception.
0x8024402C 0x8024401C 0x8024402F	These indicate network connectivity issues. Make sure your machine has network connectivity to Update Management. See the network planning section for a list of required ports and addresses.
0x8024001E	The update operation didn't complete because the service or system was shutting down.
0x8024002E	Windows Update service is disabled.
0x8024402C	If you're using a WSUS server, make sure the registry values for <code>WUserver</code> and <code>WUstatusServer</code> under the <code>HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\WindowsUpdate</code> registry key specify the correct WSUS server.
0x80072EE2	There's a network connectivity issue or an issue in talking to a configured WSUS server. Check WSUS settings and make sure the service is accessible from the client.
The service cannot be started, either because it is disabled or because it has no enabled devices associated with it. (Exception from HRESULT: 0x80070422)	Make sure the Windows Update service (<code>wuauserv</code>) is running and not disabled.
0x80070005	An access denied error can be caused by any one of the following: Infected computer Windows Update settings not configured correctly File permission error with <code>%WinDir%\SoftwareDistribution</code> folder Insufficient disk space on the system drive (C:).
Any other generic exception	Run a search on the internet for possible resolutions, and work with your local IT support.

Reviewing the `%Windir%\Windowsupdate.log` file can also help you determine possible causes. For more information about how to read the log, see [How to read the Windowsupdate.log file](#).

You can also download and run the [Windows Update troubleshooter](#) to check for any issues with Windows Update on the machine.

NOTE

The [Windows Update troubleshooter](#) documentation indicates that it's for use on Windows clients, but it also works on Windows Server.

Scenario: Update run returns Failed status (Linux)

Issue

An update run starts but encounters errors during the run.

Cause

Possible causes:

- Package manager is unhealthy.
- Update Agent (WUA for Windows, distro-specific package manager for Linux) is misconfigured.
- Specific packages are interfering with cloud-based patching.
- The machine is unreachable.
- Updates had dependencies that weren't resolved.

Resolution

If failures occur during an update run after it starts successfully, [check the job output](#) from the affected machine in the run. You might find specific error messages from your machines that you can research and take action on. Update Management requires the package manager to be healthy for successful update deployments.

If specific patches, packages, or updates are seen immediately before the job fails, you can try [excluding](#) these items from the next update deployment. To gather log information from Windows Update, see [Windows Update log files](#).

If you can't resolve a patching issue, make a copy of the `/var/opt/microsoft/omsagent/run/automationworker/omsupdategmt.log` file and preserve it for troubleshooting purposes before the next update deployment starts.

Patches aren't installed

Machines don't install updates

Try running updates directly on the machine. If the machine can't apply the updates, consult the [list of potential errors in the troubleshooting guide](#).

If updates run locally, try removing and reinstalling the agent on the machine by following the guidance at [Remove a VM from Update Management](#).

I know updates are available, but they don't show as available on my machines

This often happens if machines are configured to get updates from WSUS or Microsoft Endpoint Configuration Manager but WSUS and Configuration Manager haven't approved the updates.

You can check to see if the machines are configured for WSUS and SCCM by cross-referencing the `UseWUServer` registry key to the registry keys in the [Configuring Automatic Updates by Editing the Registry section of this article](#).

If updates aren't approved in WSUS, they're not installed. You can check for unapproved updates in Log Analytics by running the following query.

```
Update | where UpdateState == "Needed" and ApprovalSource == "WSUS" and Approved == "False" | summarize max(TimeGenerated) by Computer, KBID, Title
```

Updates show as installed, but I can't find them on my machine

Updates are often superseded by other updates. For more information, see [Update is superseded](#) in the Windows Update Troubleshooting guide.

Installing updates by classification on Linux

Deploying updates to Linux by classification ("Critical and security updates") has important caveats, especially for CentOS. These limitations are documented on the [Update Management overview page](#).

KB2267602 is consistently missing

KB2267602 is the [Windows Defender definition update](#). It's updated daily.

Next steps

If you don't see your problem or can't resolve your issue, try one of the following channels for additional support.

- Get answers from Azure experts through [Azure Forums](#).
- Connect with [@AzureSupport](#), the official Microsoft Azure account for improving customer experience.
- File an Azure support incident. Go to the [Azure support site](#) and select **Get Support**.

Troubleshoot Windows update agent issues

8/5/2021 • 5 minutes to read • [Edit Online](#)

There can be many reasons why your machine isn't showing up as ready (healthy) during an Update Management deployment. You can check the health of a Windows Hybrid Runbook Worker agent to determine the underlying problem. The following are the three readiness states for a machine:

- Ready: The Hybrid Runbook Worker is deployed and was last seen less than one hour ago.
- Disconnected: The Hybrid Runbook Worker is deployed and was last seen over one hour ago.
- Not configured: The Hybrid Runbook Worker isn't found or hasn't finished the deployment.

NOTE

There can be a slight delay between what the Azure portal shows and the current state of a machine.

This article discusses how to run the troubleshooter for Azure machines from the Azure portal, and non-Azure machines in the [offline scenario](#).

NOTE

The troubleshooter script now includes checks for Windows Server Update Services (WSUS) and for the autodownload and install keys.

Start the troubleshooter

For Azure machines, you can launch the Troubleshoot Update Agent page by selecting the **Troubleshoot** link under the **Update Agent Readiness** column in the portal. For non-Azure machines, the link brings you to this article. See [Troubleshoot offline](#) to troubleshoot a non-Azure machine.

The screenshot shows the Azure portal interface for 'TestAzureAuto - Update management'. On the left, there's a navigation sidebar with 'Process Automation' (Runbooks, Jobs, Runbooks gallery, Hybrid worker groups, Watcher tasks), 'Shared Resources' (Schedules, Modules, Modules gallery, Credentials, Connections), and a search bar. The main content area has several summary cards: 'Non-compliant machines' (3 out of 4), 'Machines need attention' (4), 'Missing updates' (44), and 'Failed update deployments' (2 out of 9 in the past six months). Below these is a table titled 'Machines (4)' with columns for MACHINE NAME, COMPLIANCE, PLATFORM, OPERATING SYSTEM, CRITICAL MISSING..., SECURITY MISSING UP..., OTHER MISSING UPDATES, and UPDATE AGENT READINESS. The rows show four machines: SQL01.internal.lab, CAS01.internal.lab, DC01.internal.lab, and LinuxVM2. LinuxVM2 is highlighted with a red border and labeled 'Disconnected (troubleshoot)' in the READINESS column.

NOTE

To check the health of the Hybrid Runbook Worker, the VM must be running. If the VM isn't running, a **Start the VM** button appears.

On the Troubleshoot Update Agent page, select **Run checks** to start the troubleshooter. The troubleshooter uses **Run Command** to run a script on the machine, to verify dependencies. When the troubleshooter is finished, it

returns the result of the checks.

The screenshot shows a window titled "Troubleshoot Update Agent (Preview)" with a sub-header "WindowsVM1". It contains a message: "Click the button below to run a troubleshooting utility in the Azure VM. This uses the RunCommand API. The results will be available in about a minute." Below this is a blue "Run checks" button with a dashed border. At the bottom left is a link "Troubleshooting documentation".

Results are shown on the page when they're ready. The checks sections show what's included in each check.

The screenshot shows a window titled "Troubleshoot Update Agent" with a sub-header "WindowsVM1". It contains a message: "Click the button below to run a troubleshooting utility in the Azure VM. This uses the RunCommand API. The results will be available in about a minute." Below this is a blue "Run checks" button with a solid border. At the bottom left is a link "Troubleshooting documentation".

The main area displays the results of various checks:

- PREREQUISITE CHECKS**
 - Operating system ✓ Passed Operating system version is supported.
 - .Net Framework 4.6.2+ ✓ Passed .NET Framework version 4.6.2+ is found.
 - WMF 5.1 ✓ Passed Detected Windows Management Framework version: 5.1.14393.2312.
 - TLS 1.2 ✓ Passed TLS 1.2 is enabled by default on the operating system.
- CONNECTIVITY CHECKS**
 - Registration endpoint ✓ Passed TCP test for 68993555-e612-4521-8149-5c92702a92b3.agentsvc.azure-automation.net (port 443) succeeded.
 - Operations endpoint ✓ Passed TCP test for eus2-jobruntimedata-prod-su1.azure-automation.net (port 443) succeeded.
- MONITORING AGENT SERVICE HEALTH CHECKS**
 - Monitoring Agent service status ✓ Passed Microsoft Monitoring Agent service (HealthService) is running.
 - Monitoring Agent service events ✓ Passed Microsoft Monitoring Agent service Event Log (Operations Manager) does not have Error event 4502 logged in the last 24 hours.
- ACCESS PERMISSION CHECKS**
 - MachineKeys folder access ✓ Passed Permissions exist to access C:\ProgramData\Microsoft\Crypto\RSA\MachineKeys.

Prerequisite checks

Operating system

The operating system check verifies whether the Hybrid Runbook Worker is running one of the operating systems shown in the next table.

OPERATING SYSTEM	NOTES
Windows Server 2012 and later	.NET Framework 4.6 or later is required. (Download the .NET Framework .) Windows PowerShell 5.1 is required. (Download Windows Management Framework 5.1 .)

.NET 4.6.2

The .NET Framework check verifies that the system has .NET Framework 4.6.2 or later installed.

WMF 5.1

The WMF check verifies that the system has the required version of the Windows Management Framework (WMF), which is [Windows Management Framework 5.1](#).

TLS 1.2

This check determines whether you're using TLS 1.2 to encrypt your communications. TLS 1.0 is no longer supported by the platform. Use TLS 1.2 to communicate with Update Management.

Connectivity checks

Registration endpoint

This check determines whether the agent can properly communicate with the agent service.

Proxy and firewall configurations must allow the Hybrid Runbook Worker agent to communicate with the registration endpoint. For a list of addresses and ports to open, see [Network planning](#).

Operations endpoint

This check determines whether the agent can properly communicate with the Job Runtime Data Service.

Proxy and firewall configurations must allow the Hybrid Runbook Worker agent to communicate with the Job Runtime Data Service. For a list of addresses and ports to open, see [Network planning](#).

VM service health checks

Monitoring agent service status

This check determines if the Log Analytics agent for Windows (`healthservice`) is running on the machine. To learn more about troubleshooting the service, see [The Log Analytics agent for Windows isn't running](#).

To reinstall the Log Analytics agent for Windows, see [Install the agent for Windows](#).

Monitoring agent service events

This check determines whether any 4502 events appear in the Azure Operations Manager log on the machine in the past 24 hours.

To learn more about this event, see the [Event 4502 in the Operations Manager log](#) for this event.

Access permissions checks

NOTE

The troubleshooter currently doesn't route traffic through a proxy server if one is configured.

Crypto folder access

The Crypto folder access check determines whether the local system account has access to `C:\ProgramData\Microsoft\Crypto\RSA`.

Troubleshoot offline

You can use the troubleshooter on a Hybrid Runbook Worker offline by running the script locally. Get the following script from GitHub: [UM_Windows_Troubleshooter_Offline.ps1](#). To run the script, you must have WMF 4.0 or later installed. To download the latest version of PowerShell, see [Installing various versions of PowerShell](#).

The output of this script looks like the following example:

```

RuleId : OperatingSystemCheck
RuleGroupId : prerequisites
RuleName : Operating System
RuleGroupName : Prerequisite Checks
RuleDescription : The Windows Operating system must be version 6.2.9200 (Windows Server 2012) or higher
CheckResult : Passed
CheckResultMessage : Operating System version is supported
CheckResultMessageId : OperatingSystemCheck.Passed
CheckResultMessageArguments : {}

RuleId : DotNetFrameworkInstalledCheck
RuleGroupId : prerequisites
RuleName : .NET Framework 4.5+
RuleGroupName : Prerequisite Checks
RuleDescription : .NET Framework version 4.5 or higher is required
CheckResult : Passed
CheckResultMessage : .NET Framework version 4.5+ is found
CheckResultMessageId : DotNetFrameworkInstalledCheck.Passed
CheckResultMessageArguments : {}

RuleId : WindowsManagementFrameworkInstalledCheck
RuleGroupId : prerequisites
RuleName : WMF 5.1
RuleGroupName : Prerequisite Checks
RuleDescription : Windows Management Framework version 4.0 or higher is required (version 5.1 or higher is preferable)
CheckResult : Passed
CheckResultMessage : Detected Windows Management Framework version: 5.1.17763.1
CheckResultMessageId : WindowsManagementFrameworkInstalledCheck.Passed
CheckResultMessageArguments : {5.1.17763.1}

RuleId : AutomationAgentServiceConnectivityCheck1
RuleGroupId : connectivity
RuleName : Registration endpoint
RuleGroupName : connectivity
RuleDescription :
CheckResult : Failed
CheckResultMessage : Unable to find Workspace registration information in registry
CheckResultMessageId : AutomationAgentServiceConnectivityCheck1.Failed.NoRegistrationFound
CheckResultMessageArguments : {}

RuleId : AutomationJobRuntimeDataServiceConnectivityCheck
RuleGroupId : connectivity
RuleName : Operations endpoint
RuleGroupName : connectivity
RuleDescription : Proxy and firewall configuration must allow Automation Hybrid Worker agent to communicate with eus2-jobruntimedata-prod-su1.azure-automation.net
CheckResult : Passed
CheckResultMessage : TCP Test for eus2-jobruntimedata-prod-su1.azure-automation.net (port 443) succeeded
CheckResultMessageId : AutomationJobRuntimeDataServiceConnectivityCheck.Passed
CheckResultMessageArguments : {eus2-jobruntimedata-prod-su1.azure-automation.net}

RuleId : MonitoringAgentServiceRunningCheck
RuleGroupId : servicehealth
RuleName : Monitoring Agent service status
RuleGroupName : VM Service Health Checks
RuleDescription : HealthService must be running on the machine
CheckResult : Failed
CheckResultMessage : Log Analytics for Windows service (HealthService) is not running
CheckResultMessageId : MonitoringAgentServiceRunningCheck.Failed
CheckResultMessageArguments : {Log Analytics agent for Windows, HealthService}

RuleId : MonitoringAgentServiceEventsCheck
RuleGroupId : servicehealth
RuleName : Monitoring Agent service events
RuleGroupName : VM Service Health Checks
RuleDescription : Event Log must not have event 4502 logged in the past 24 hours

```

```
CheckResult           : Failed
CheckResultMessage   : Log Analytics agent for Windows service Event Log (Operations Manager) does
not exist on the machine
CheckResultMessageId : MonitoringAgentServiceEventsCheck.Failed.NoLog
CheckResultMessageArguments : {Log Analytics agent for Windows, Operations Manager, 4502}

RuleId               : CryptoRsaMachineKeysFolderAccessCheck
RuleGroupId          : permissions
RuleName             : Crypto RSA MachineKeys Folder Access
RuleGroupName         : Access Permission Checks
RuleDescription       : SYSTEM account must have WRITE and MODIFY access to
'C:\\\\ProgramData\\\\Microsoft\\\\Crypto\\\\RSA\\\\MachineKeys'
CheckResult          : Passed
CheckResultMessage   : Have permissions to access
C:\\\\ProgramData\\\\Microsoft\\\\Crypto\\\\RSA\\\\MachineKeys
CheckResultMessageId : CryptoRsaMachineKeysFolderAccessCheck.Passed
CheckResultMessageArguments : {C:\\\\ProgramData\\\\Microsoft\\\\Crypto\\\\RSA\\\\MachineKeys}

RuleId               : TlsVersionCheck
RuleGroupId          : prerequisites
RuleName             : TLS 1.2
RuleGroupName         : Prerequisite Checks
RuleDescription       : Client and Server connections must support TLS 1.2
CheckResult          : Passed
CheckResultMessage   : TLS 1.2 is enabled by default on the Operating System.
CheckResultMessageId : TlsVersionCheck.Passed.EnabledByDefault
CheckResultMessageArguments : {}
```

Next steps

[Troubleshoot Hybrid Runbook Worker issues.](#)

Troubleshoot Linux update agent issues

8/13/2021 • 4 minutes to read • [Edit Online](#)

There can be many reasons why your machine isn't showing up as ready (healthy) in Update Management. You can check the health of a Linux Hybrid Runbook Worker agent to determine the underlying problem. The following are the three readiness states for a machine:

- Ready: The Hybrid Runbook Worker is deployed and was last seen less than one hour ago.
- Disconnected: The Hybrid Runbook Worker is deployed and was last seen over one hour ago.
- Not configured: The Hybrid Runbook Worker isn't found or hasn't finished deployment.

NOTE

There can be a slight delay between what the Azure portal shows and the current state of a machine.

This article discusses how to run the troubleshooter for Azure machines from the Azure portal and non-Azure machines in the [offline scenario](#).

NOTE

The troubleshooter script currently doesn't route traffic through a proxy server if one is configured.

Start the troubleshooter

For Azure machines, select the **troubleshoot** link under the **Update Agent Readiness** column in the portal to open the Troubleshoot Update Agent page. For non-Azure machines, the link brings you to this article. To troubleshoot a non-Azure machine, see the instructions in the "Troubleshoot offline" section.

The screenshot shows the Azure portal interface for a machine named 'TestAzureAuto - Update management'. The main dashboard displays various metrics: Non-compliant machines (3 out of 4), Machines need attention (4), Missing updates (44), and Failed update deployments (2 out of 9 in the past six months). Below this, a table lists four machines: SQL01.internal.lab, CAS01.internal.lab, DCO1.internal.lab, and LinuxVM2. The table includes columns for Machine Name, Compliance (Status), Platform, Operating System, Critical Missing Updates, Security Missing Updates, Other Missing Updates, and Update Agent Readiness. The 'LinuxVM2' row shows a status of 'Compliant' with a green checkmark, but 'Disconnected' with a red exclamation mark in the 'Update Agent Readiness' column, which is highlighted with a red border.

NOTE

The checks require the VM to be running. If the VM isn't running, **Start the VM** appears.

On the Troubleshoot Update Agent page, select **Run Checks** to start the troubleshooter. The troubleshooter uses [Run command](#) to run a script on the machine to verify the dependencies. When the troubleshooter is finished, it returns the result of the checks.

 Troubleshoot Update Agent (Preview)

LinuxVM2

Click the button below to run a troubleshooting utility in the Azure VM. [This uses the RunCommand API.](#)
The results will be available in about a minute.

Run checks

[Troubleshooting documentation](#)

When the checks are finished, the results are returned in the window. The check sections provide information on what each check is looking for.

 Troubleshoot Update Agent (Preview)

LinuxVM2

Click the button below to run a troubleshooting utility in the Azure VM. [This uses the RunCommand API.](#)
The results will be available in about a minute.

Run checks

[Troubleshooting documentation](#)

 PREREQUISITE CHECKS

Operating system 	 Passed	Operating system version is supported.
--	--	--

 MONITORING AGENT SERVICE HEALTH CHECKS

OMS agent 	 Passed	OMS (Operations Management Suite) agent is installed.
OMS agent status 	 Passed	OMS (Operations Management Suite) agent is running.
Multihoming 	 Passed	Machine registered with Log Analytics workspace: ['68993555-e612-4521-8149-5c92702a92b3'].
Hybrid runbook worker 	 Passed	Hybrid runbook worker package is present.
Hybrid runbook worker status 	 Failed	Hybrid runbook worker is not running. current_mof file: (/etc/opt/omi/conf/omsconfig/configuration/Current.mof) is missing.

 CONNECTIVITY CHECKS

General internet connectivity 	 Passed	Machine is connected to internet.
Registration endpoint 	 Passed	TCP test for scus-agentservice-prod-1.azure-automation.net (port 443) succeeded.
Operations endpoint 	 Passed	TCP test for eus2-jobruntimedata-prod-su1.azure-automation.net (port 443) succeeded.
Log Analytics endpoint 1 	 Passed	TCP test for 68993555-e612-4521-8149-5c92702a92b3.ods.opinsights.azure.com (port 443) succeeded.
Log Analytics endpoint 2 	 Passed	TCP test for 68993555-e612-4521-8149-5c92702a92b3.oms.opinsights.azure.com (port 443) succeeded.
Log Analytics endpoint 3 	 Passed	TCP test for ods.systemcenteradvisor.com (port 443) succeeded.

Prerequisite checks

Operating system

The operating system check verifies if the Hybrid Runbook Worker is running one of the following operating systems.

OPERATING SYSTEM	NOTES
CentOS 6 (x86/x64) and 7 (x64)	Linux agents must have access to an update repository. Classification-based patching requires 'yum' to return security data, which CentOS doesn't have out of the box.

OPERATING SYSTEM	NOTES
Red Hat Enterprise 6 (x86/x64) and 7 (x64)	Linux agents must have access to an update repository.
SUSE Linux Enterprise Server 11 (x86/x64) and 12 (x64)	Linux agents must have access to an update repository.
Ubuntu 14.04 LTS, 16.04 LTS, and 18.04 LTS (x86/x64)	Linux agents must have access to an update repository.

Monitoring agent service health checks

Log Analytics agent

This check ensures that the Log Analytics agent for Linux is installed. For instructions on how to install it, see [Install the agent for Linux](#).

Log Analytics agent status

This check ensures that the Log Analytics agent for Linux is running. If the agent isn't running, you can run the following command to attempt to restart it. For more information on troubleshooting the agent, see [Linux - Troubleshoot Hybrid Runbook Worker issues](#).

```
sudo /opt/microsoft/omsagent/bin/service_control restart
```

Multihoming

This check determines if the agent is reporting to multiple workspaces. Update Management doesn't support multihoming.

Hybrid Runbook Worker

This check verifies if the Log Analytics agent for Linux has the Hybrid Runbook Worker package. This package is required for Update Management to work. To learn more, see [Log Analytics agent for Linux isn't running](#).

Update Management downloads Hybrid Runbook Worker packages from the operations endpoint. Therefore, if the Hybrid Runbook Worker is not running and the [operations endpoint](#) check fails, the update can fail.

Hybrid Runbook Worker status

This check makes sure the Hybrid Runbook Worker is running on the machine. The processes in the example below should be present if the Hybrid Runbook Worker is running correctly.

```
nxautom+ 8567      1 0 14:45 ?      00:00:00 python
/opt/microsoft/omsconfig/modules/nxOMSAutomationWorker/DSCResources/MSFT_nxOMSAutomationWorkerResource/automationworker/worker/main.py /var/opt/microsoft/omsagent/state/automationworker/oms.conf rworkspace:<workspaceId> <Linux hybrid worker version>
nxautom+ 8593      1 0 14:45 ?      00:00:02 python
/opt/microsoft/omsconfig/modules/nxOMSAutomationWorker/DSCResources/MSFT_nxOMSAutomationWorkerResource/automationworker/worker/hybridworker.py /var/opt/microsoft/omsagent/state/automationworker/worker.conf managed rworkspace:<workspaceId> rversion:<Linux hybrid worker version>
nxautom+ 8595      1 0 14:45 ?      00:00:02 python
/opt/microsoft/omsconfig/modules/nxOMSAutomationWorker/DSCResources/MSFT_nxOMSAutomationWorkerResource/automationworker/worker/hybridworker.py
/var/opt/microsoft/omsagent/<workspaceId>/state/automationworker/diy/worker.conf managed rworkspace:<workspaceId> rversion:<Linux hybrid worker version>
```

Connectivity checks

General internet connectivity

This check makes sure that the machine has access to the internet.

Registration endpoint

This check determines if the Hybrid Runbook Worker can properly communicate with Azure Automation in the Log Analytics workspace.

Proxy and firewall configurations must allow the Hybrid Runbook Worker agent to communicate with the registration endpoint. For a list of addresses and ports to open, see [Network planning](#).

Operations endpoint

This check determines if the Log Analytics agent can properly communicate with the Job Runtime Data Service.

Proxy and firewall configurations must allow the Hybrid Runbook Worker agent to communicate with the Job Runtime Data Service. For a list of addresses and ports to open, see [Network planning](#).

Log Analytics endpoint 1

This check verifies that your machine has access to the endpoints needed by the Log Analytics agent.

Log Analytics endpoint 2

This check verifies that your machine has access to the endpoints needed by the Log Analytics agent.

Log Analytics endpoint 3

This check verifies that your machine has access to the endpoints needed by the Log Analytics agent.

Troubleshoot offline

You can use the troubleshooter offline on a Hybrid Runbook Worker by running the script locally. The Python script, [UM_Linux_Troubleshooter_Offline.py](#), can be found in GitHub. An example of the output of this script is shown in the following example:

```

Debug: Machine Information: Static hostname: LinuxVM2
      Icon name: computer-vm
      Chassis: vm
      Machine ID: 00000000000000000000000000000000
      Boot ID: 00000000000000000000000000000000
      Virtualization: microsoft
      Operating System: Ubuntu 16.04.5 LTS
      Kernel: Linux 4.15.0-1025-azure
      Architecture: x86-64

Passed: Operating system version is supported

Passed: Microsoft Monitoring agent is installed

Debug: omsadmin.conf file contents:
      WORKSPACE_ID=00000000-0000-0000-0000-000000000000
      AGENT_GUID=00000000-0000-0000-0000-000000000000
      LOG_FACILITY=local0
      CERTIFICATE_UPDATE_ENDPOINT=https://00000000-0000-0000-0000-
000000000000.oms.opinsights.azure.com/ConfigurationService.Svc/RenewCertificate
      URL_TLD=opinsights.azure.com
      DSC_ENDPOINT=https://scus-agentservice-prod-1.azure-automation.net/Account
0000-0000-0000-000000000000/Nodes\(\AgentId='00000000-0000-0000-0000-000000000000'\)\
      OMS_ENDPOINT=https://00000000-0000-0000-0000-000000000000.ods.opinsights
.azure.com/OperationalData.svc/PostJsonDataItems
      AZURE_RESOURCE_ID=/subscriptions/00000000-0000-0000-0000-000000000000/re
sourcegroups/myresourcegroup/providers/microsoft.compute/virtualmachines/linuxvm
      OMSCLOUD_ID=0000-0000-0000-0000-0000-00
      UUID=00000000-0000-0000-000000000000

Passed: Microsoft Monitoring agent is running

Passed: Machine registered with log analytics workspace:['00000000-0000-0000-0000-000000000000']

Passed: Hybrid worker package is present

Passed: Hybrid worker is running

Passed: Machine is connected to internet

Passed: TCP test for {scus-agentservice-prod-1.azure-automation.net} (port 443) succeeded

Passed: TCP test for {eus2-jobruntimedata-prod-su1.azure-automation.net} (port 4
43) succeeded

Passed: TCP test for {00000000-0000-0000-0000-000000000000.ods.opinsights.azure.
succeeded

Passed: TCP test for {00000000-0000-0000-0000-000000000000.oms.opinsights.azure.
com} (port 443) succeeded

Passed: TCP test for {ods.systemcenteradvisor.com} (port 443) succeeded

```

Next steps

[Troubleshoot Hybrid Runbook Worker issues.](#)

Send an email from a runbook

8/9/2021 • 3 minutes to read • [Edit Online](#)

You can send an email from a runbook with [SendGrid](#) using PowerShell.

Prerequisites

- Azure subscription. If you don't have one yet, you can [activate your MSDN subscriber benefits](#) or sign up for a [free account](#).
- [A SendGrid account](#).
- Sender Verification has been configured in Send Grid. Either [Domain or Single Sender](#)
- [Automation account](#) with [Az modules](#).
- [Run As account](#) to store and execute the runbook.

Create an Azure Key Vault

You can create an Azure Key Vault using the following PowerShell script. Replace the variable values with values specific to your environment. Use the embedded Azure Cloud Shell via the **Try It** button, located in the top-right corner of the code block. You can also copy and run the code locally if you have the [Az modules](#) installed on your local machine. This script also creates a [Key Vault access policy](#) that allows the Run As account to get and set key vault secrets in the specified key vault.

NOTE

To retrieve your API key, use the steps in [Find your SendGrid API key](#).

```

$SubscriptionId = "<subscription ID>"

# Sign in to your Azure account and select your subscription
# If you omit the SubscriptionId parameter, the default subscription is selected.
Connect-AzAccount -SubscriptionId $SubscriptionId

# Use Get-AzLocation to see your available locations.
$region = "southcentralus"
$keyVaultResourceGroupName = "mykeyvaultgroup"
$vaultName = "<Enter a universally unique vault name>"
$SendGridAPIKey = "<SendGrid API key>"
$automationAccountName = "testaa"

# Create new Resource Group, or omit this step if you already have a resource group.
New-AzResourceGroup -Name $keyVaultResourceGroupName -Location $region

# Create the new key vault
$newKeyVault = New-AzKeyVault -VaultName $vaultName -ResourceGroupName $keyVaultResourceGroupName -Location $region
$resourceId = $newKeyVault.ResourceId

# Convert the SendGrid API key into a SecureString
$secret = ConvertTo-SecureString -String $SendGridAPIKey -AsPlainText -Force
Set-AzKeyVaultSecret -VaultName $VaultName -Name 'SendGridAPIKey' -SecretValue $secret

# Grant access to the Key Vault to the Automation Run As account.
$connection = Get-AzAutomationConnection -ResourceGroupName $keyVaultResourceGroupName -
AutomationAccountName $automationAccountName -Name AzureRunAsConnection
$appID = $connection.FieldDefinitionValues.ApplicationId
Set-AzKeyVaultAccessPolicy -VaultName $VaultName -ServicePrincipalName $appID -PermissionsToSecrets Set, Get

```

For other ways to create an Azure Key Vault and store a secret, see [Key Vault quickstarts](#).

Import required modules into your Automation account

To use Azure Key Vault within a runbook, you must import the following modules into your Automation account:

- [Az.Accounts](#)
- [Az.KeyVault](#)

For instructions, see [Import Az modules](#).

Create the runbook to send an email

After you have created a Key Vault and stored your `SendGrid` API key, it's time to create the runbook that retrieves the API key and sends an email. Let's use a runbook that uses `AzureRunAsConnection` as a [Run As account](#) to authenticate with Azure to retrieve the secret from Azure Key Vault. We'll call the runbook **Send-GridMailMessage**. You can modify the PowerShell script used for example purposes, and reuse it for different scenarios.

1. Go to your Azure Automation account.
2. Under **Process Automation**, select **Runbooks**.
3. At the top of the list of runbooks, select + **Create a runbook**.
4. On the Add Runbook page, enter **Send-GridMailMessage** for the runbook name. For the runbook type, select **PowerShell**. Then, select **Create**.

The screenshot shows the Azure portal interface for an Automation Account named 'testaa'. On the left, there's a navigation sidebar with various options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Configuration Management, Inventory, Change tracking, State configuration (DSC), Update management, and Process Automation. The main area displays a list of existing runbooks:

NAME	AUTHORING STATUS	RUNBOOK TYPE
AzureAutomationTuto...	✓ Published	Graphical Runbook
AzureAutomationTuto...	✓ Published	Python 2 Runbook
AzureAutomationTuto...	✓ Published	PowerShell Runbo...
AzureClassicAutomati...	✓ Published	Graphical Runbook
AzureClassicAutomati...	✓ Published	PowerShell Runbo...

To the right, a modal window titled 'Create a runbook' is open. It has fields for 'Name' (set to 'Send-GridMailMessage'), 'Runbook type' (set to 'PowerShell'), and a 'Description' field which is empty. A 'Create' button is at the bottom.

5. The runbook is created and the Edit PowerShell Runbook page opens.

The screenshot shows the 'Edit PowerShell Runbook' page for a runbook named 'Send-GridMailMessage'. The top navigation bar includes Save, Publish, Revert to published, Test pane, and Feedback buttons. On the left, there's a sidebar with three sections: 'CMDLETS', 'RUNBOOKS', and 'ASSETS'. The main area is currently empty, showing a single numbered item '1'.

6. Copy the following PowerShell example into the Edit page. Ensure that the `VaultName` specifies the name you've chosen for your Key Vault.

```

Param(
    [Parameter(Mandatory=$True)]
    [String] $destEmailAddress,
    [Parameter(Mandatory=$True)]
    [String] $fromEmailAddress,
    [Parameter(Mandatory=$True)]
    [String] $subject,
    [Parameter(Mandatory=$True)]
    [String] $content
)

$Conn = Get-AutomationConnection -Name AzureRunAsConnection
Connect-AzAccount -ServicePrincipal -Tenant $Conn.TenantID -ApplicationId $Conn.ApplicationID -
CertificateThumbprint $Conn.CertificateThumbprint | Out-Null
$VaultName = "<Enter your vault name>"
$SENDGRID_API_KEY = Get-AzKeyVaultSecret -VaultName $VaultName -Name "SendGridAPIKey" -AsPlainText
$headers = New-Object "System.Collections.Generic.Dictionary[[String],[String]]"
$headers.Add("Authorization", "Bearer " + $SENDGRID_API_KEY)
$headers.Add("Content-Type", "application/json")

$body = @{
personalizations = @(
    @{
        to = @(
            @{
                email = $destEmailAddress
            }
        )
    }
)
from = @{
    email = $fromEmailAddress
}
subject = $subject
content = @(
    @{
        type = "text/plain"
        value = $content
    }
)
}

$bodyJson = $body | ConvertTo-Json -Depth 4

$response = Invoke-RestMethod -Uri https://api.sendgrid.com/v3/mail/send -Method Post -Headers
$headers -Body $bodyJson

```

7. Select Publish to save and publish the runbook.

To verify that the runbook executes successfully, you can follow the steps under [Test a runbook](#) or [Start a runbook](#).

If you don't initially see your test email, check your **Junk** and **Spam** folders.

Clean up resources after the email operation

- When the runbook is no longer needed, select it in the runbook list and click **Delete**.
- Delete the Key Vault by using the [Remove-AzKeyVault](#) cmdlet.

```

$VaultName = "<your KeyVault name>"
$ResourceGroupName = "<your ResourceGroup name>"
Remove-AzKeyVault -VaultName $VaultName -ResourceGroupName $ResourceGroupName

```

Next steps

- To send runbook job data to your Log Analytics workspace, see [Forward Azure Automation job data to Azure Monitor logs](#).
- To monitor base-level metrics and logs, see [Use an alert to trigger an Azure Automation runbook](#).
- To correct issues arising during runbook operations, see [Troubleshoot runbook issues](#).

Monitor runbooks with metric alerts

3/5/2021 • 3 minutes to read • [Edit Online](#)

In this article, you learn how to create a [metric alert](#) based on runbook completion status.

Sign in to Azure

Sign in to the [Azure portal](#)

Create alert

Alerts allow you to define a condition to monitor for and an action to take when that condition is met.

1. Launch the Azure Automation service in the Azure portal by clicking **All services**, then searching for and selecting **Automation Accounts**.
2. In your list of Automation accounts, select the account you want to create an alert for.
3. Under **Monitoring**, select **Alerts** and then select **+ New Alert Rule**. The scope for the target is already defined and associated with your Automation account.

Configure alert criteria

1. Click **Select Condition**. Select **Metrics** for the **Signal type**, and choose **Total Jobs** from the list.
2. The **Configure signal logic** page is where you define the logic that triggers the alert. Under the historical graph you are presented with two dimensions, **Runbook Name** and **Status**. Dimensions are different properties for a metric that can be used to filter results. For **Runbook Name**, select the runbook you want to alert on or leave blank to alert on all runbooks. For **Status**, select a status from the drop-down you want to monitor for. The runbook name and status values that appear in the dropdown are only for jobs that have ran in the past week.

If you want to alert on a status or runbook that isn't shown in the dropdown, click the **Add custom value** option next to the dimension. This action opens a dialog that allows you to specify a custom value, which hasn't emitted for that dimension recently. If you enter a value that doesn't exist for a property your alert won't be triggered.

NOTE

If you don't specify a name for the **Runbook Name** dimension, if there are any runbooks that meet the status criteria, which includes hidden system runbooks, you will receive an alert.

For example, to alert when a runbook returns a *Failed* status, in addition to specifying the runbook name, for the **Status** dimension add the custom dimension value **Failed**.

This metric supports dimensions. Selecting the dimension values will help you filter to the right time series. If you do not select any value for a dimension, that dimension will be ignored. ⓘ

Dimension name	Operator	Dimension values
Runbook Name	=	ScheduledStartStop_Child
Status	=	<input type="text"/> Add custom value

In an alert rule with multiple conditions, you can only select one dimension value per condition. Selected dimension values will be applied to all conditions.

Alert logic

Add custom dimension value

Specify custom values which are not yet emitted

Status: Failed

OK Cancel

Done

3. Under **Alert logic**, define the condition and threshold for your alert. A preview of your condition defined is shown underneath.
4. Under **Evaluated based on**, select the timespan for the query and how often you want that query to run. For example, if you choose **Over the last 5 minutes** for **Period**, and **Every 1 Minute** for **Frequency**, the alert looks for the number of runbooks that met your criteria over the past 5 minutes. This query is run every minute, and once the alert criteria you defined is no longer found in a 5-minute window, the alert resolves itself. When finished, click **Done**.

Configure signal logic

This metric supports dimensions. Selecting the dimension values will help you filter to the right time series. If you do not select any value for a dimension, that dimension will be ignored.

DIMENSION NAME	DIMENSION VALUES	SELECT *
Runbook Name	AzureAutomationTutorialScript	<input type="button"/> <input type="button"/>
Status	Failed	<input type="button"/> <input type="button"/>

Alert logic

Condition: Greater than Time Aggregation: Total Threshold: 0 count

Condition preview: Whenever the total jobs is greater than 0 count

Evaluated based on

Period (grain): Over the last 5 minutes Frequency: Every 1 Minute

Done

Define the action to take

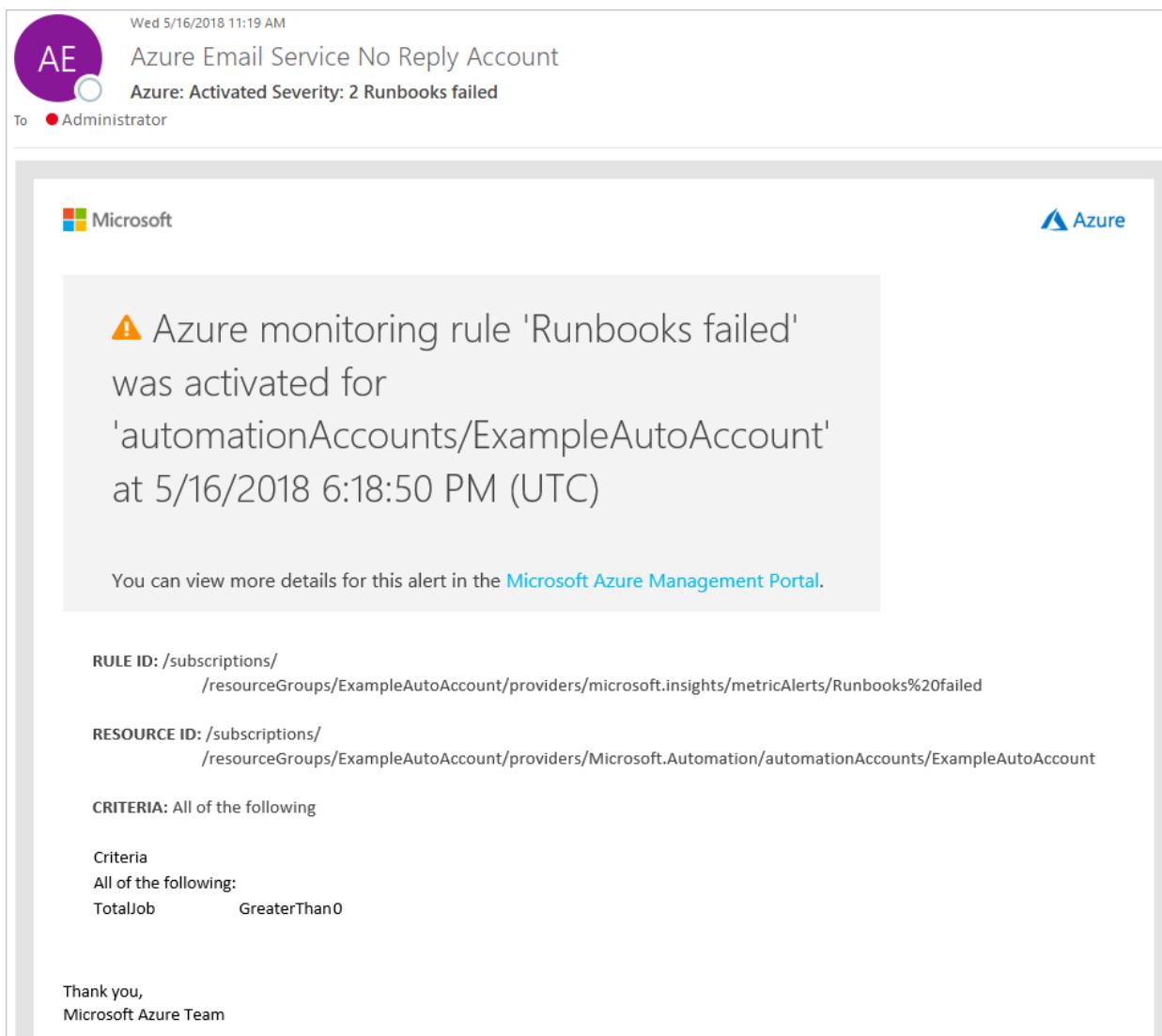
1. Under **Action group**, select **Specify action group**. An action group is a group of actions that you can use across more than one alert. These can include but aren't limited to, email notifications, runbooks, webhooks, and many more. To learn more about action groups and steps to create one that sends an email notification, see [Create and manage action groups](#).

Define alert details

1. Under **Alert rule details**, give the alert a friendly name and description. Set the **Severity** to match your alert condition. There are five severities ranging from 0 to 5. The alerts are treated the same independent of the severity, you can match the severity to match your business logic.
2. By default rules are enabled at creation, unless you select **No** for the option **Enable alert rule upon creation**. For alerts created in a disabled state, you can enable them in the future when you are ready. Select **Create alert rule** to save your changes.

Receive notification

When the alert criteria is met, the action group runs the action defined. In this article's example, an email is sent. The following image is an example of an email you receive after the alert is triggered:



Once the metric is no longer outside of the threshold defined, the alert is deactivated and the action group runs the action defined. If an email action type is selected, a resolution email is sent stating it has been resolved.

Next steps

- For more information, see [Use an alert to trigger an Azure Automation runbook](#).

Use an alert to trigger an Azure Automation runbook

4/22/2021 • 6 minutes to read • [Edit Online](#)

You can use [Azure Monitor](#) to monitor base-level metrics and logs for most services in Azure. You can call Azure Automation runbooks by using [action groups](#) to automate tasks based on alerts. This article shows you how to configure and run a runbook by using alerts.

Alert types

You can use automation runbooks with three alert types:

- Common alerts
- Activity log alerts
- Near-real-time metric alerts

NOTE

The common alert schema standardizes the consumption experience for alert notifications in Azure today. Historically, the three alert types in Azure today (metric, log, and activity log) have had their own email templates, webhook schemas, etc. To learn more, see [Common alert schema](#)

When an alert calls a runbook, the actual call is an HTTP POST request to the webhook. The body of the POST request contains a JSON-formatted object that has useful properties that are related to the alert. The following table lists links to the payload schema for each alert type:

ALERT	DESCRIPTION	PAYOUT SCHEMA
Common alert	The common alert schema that standardizes the consumption experience for alert notifications in Azure today.	Common alert payload schema
Activity log alert	Sends a notification when any new event in the Azure activity log matches specific conditions. For example, when a <code>Delete VM</code> operation occurs in <code>myProductionResourceGroup</code> or when a new Azure Service Health event with an Active status appears.	Activity log alert payload schema
Near real-time metric alert	Sends a notification faster than metric alerts when one or more platform-level metrics meet specified conditions. For example, when the value for <code>CPU %</code> on a VM is greater than 90, and the value for <code>Network In</code> is greater than 500 MB for the past 5 minutes.	Near real-time metric alert payload schema

Because the data that's provided by each type of alert is different, each alert type is handled differently. In the next section, you learn how to create a runbook to handle different types of alerts.

Create a runbook to handle alerts

To use Automation with alerts, you need a runbook that has logic that manages the alert JSON payload that's passed to the runbook. The following example runbook must be called from an Azure alert.

As described in the preceding section, each type of alert has a different schema. The script takes the webhook data from an alert in the `WebhookData` runbook input parameter. Then, the script evaluates the JSON payload to determine which alert type is being used.

This example uses an alert from a VM. It retrieves the VM data from the payload, and then uses that information to stop the VM. The connection must be set up in the Automation account where the runbook is run. When using alerts to trigger runbooks, it is important to check the alert status in the runbook that is triggered. The runbook triggers each time the alert changes state. Alerts have multiple states, with the two most common being Activated and Resolved. Check for state in your runbook logic to ensure that the runbook does not run more than once. The example in this article shows how to look for alerts with state Activated only.

The runbook uses the connection asset `AzureRunAsConnection` [Run As account](#) to authenticate with Azure to perform the management action against the VM.

Use this example to create a runbook called `Stop-AzureVmInResponsetoVMAlert`. You can modify the PowerShell script, and use it with many different resources.

1. Go to your Azure Automation account.
2. Under **Process Automation**, select **Runbooks**.
3. At the top of the list of runbooks, select + **Create a runbook**.
4. On the **Add Runbook** page, enter `Stop-AzureVmInResponsetoVMAlert` for the runbook name. For the runbook type, select **PowerShell**. Then, select **Create**.
5. Copy the following PowerShell example into the **Edit** page.

```
[OutputType("PSAzureOperationResponse")]
param
(
    [Parameter (Mandatory=$false)]
    [object] $WebhookData
)
$ErrorActionPreference = "stop"

if ($WebhookData)
{
    # Get the data object from WebhookData
    $WebhookBody = (ConvertFrom-Json -InputObject $WebhookData.RequestBody)

    # Get the info needed to identify the VM (depends on the payload schema)
    $schemaId = $WebhookBody.schemaId
    Write-Verbose "schemaId: $schemaId" -Verbose
    if ($schemaId -eq "azureMonitorCommonAlertSchema") {
        # This is the common Metric Alert schema (released March 2019)
        $Essentials = [object] ($WebhookBody.data).essentials
        # Get the first target only as this script doesn't handle multiple
        $alertTargetIdArray = (($Essentials.alertTargetIds)[0]).Split("/")
        $SubId = ($alertTargetIdArray)[2]
        $ResourceGroupName = ($alertTargetIdArray)[4]
        $ResourceType = ($alertTargetIdArray)[6] + "/" + ($alertTargetIdArray)[7]
        $ResourceName = ($alertTargetIdArray)[-1]
        $status = $Essentials.monitorCondition
    }
    elseif ($schemaId -eq "AzureMonitorMetricAlert") {
        # This is the near-real-time Metric Alert schema
        $AlertContext = [object] ($WebhookBody.data).context
    }
}
```

```

$SubId = $AlertContext.subscriptionId
$ResourceGroupName = $AlertContext.resourceGroupName
$ResourceType = $AlertContext.resourceType
$ResourceName = $AlertContext.resourceName
$status = ($WebhookBody.data).status
}
elseif ($schemaId -eq "Microsoft.Insights/activityLogs") {
    # This is the Activity Log Alert schema
    $AlertContext = [object] ((($WebhookBody.data).context).activityLog
    $SubId = $AlertContext.subscriptionId
    $ResourceGroupName = $AlertContext.resourceGroupName
    $ResourceType = $AlertContext.resourceType
    $ResourceName = (($AlertContext.resourceId).Split("/"))[-1]
    $status = ($WebhookBody.data).status
}
elseif ($schemaId -eq $null) {
    # This is the original Metric Alert schema
    $AlertContext = [object] $WebhookBody.context
    $SubId = $AlertContext.subscriptionId
    $ResourceGroupName = $AlertContext.resourceGroupName
    $ResourceType = $AlertContext.resourceType
    $ResourceName = $AlertContext.resourceName
    $status = $WebhookBody.status
}
else {
    # Schema not supported
    Write-Error "The alert data schema - $schemaId - is not supported."
}

Write-Verbose "status: $status" -Verbose
if (($status -eq "Activated") -or ($status -eq "Fired"))
{
    Write-Verbose "resourceType: $ResourceType" -Verbose
    Write-Verbose "resourceName: $ResourceName" -Verbose
    Write-Verbose "resourceGroupName: $ResourceGroupName" -Verbose
    Write-Verbose "subscriptionId: $SubId" -Verbose

    # Determine code path depending on the resourceType
    if ($ResourceType -eq "Microsoft.Compute/virtualMachines")
    {
        # This is an Resource Manager VM
        Write-Verbose "This is an Resource Manager VM." -Verbose

        # Authenticate to Azure with service principal and certificate and set subscription
        Write-Verbose "Authenticating to Azure with service principal and certificate" -Verbose
        $ConnectionAssetName = "AzureRunAsConnection"
        Write-Verbose "Get connection asset: $ConnectionAssetName" -Verbose
        $Conn = Get-AutomationConnection -Name $ConnectionAssetName
        if ($Conn -eq $null)
        {
            throw "Could not retrieve connection asset: $ConnectionAssetName. Check that this asset exists in the Automation account."
        }
        Write-Verbose "Authenticating to Azure with service principal." -Verbose
        Add-AzAccount -ServicePrincipal -Tenant $Conn.TenantID -ApplicationId $Conn.ApplicationID -CertificateThumbprint $Conn.CertificateThumbprint | Write-Verbose
        Write-Verbose "Setting subscription to work against: $SubId" -Verbose
        Set-AzContext -SubscriptionId $SubId -ErrorAction Stop | Write-Verbose

        # Stop the Resource Manager VM
        Write-Verbose "Stopping the VM - $ResourceName - in resource group - $ResourceGroupName -" -Verbose
        Stop-AzVM -Name $ResourceName -ResourceGroupName $ResourceGroupName -Force
        # [OutputType(PSAzureOperationResponse")]
    }
    else {
        # ResourceType not supported
        Write-Error "$ResourceType is not a supported resource type for this runbook."
    }
}

```

```

    }
    else {
        # The alert status was not 'Activated' or 'Fired' so no action taken
        Write-Verbose ("No action taken. Alert status: " + $status) -Verbose
    }
}
else {
    # Error
    Write-Error "This runbook is meant to be started from an Azure alert webhook only."
}

```

6. Select **Publish** to save and publish the runbook.

Create the alert

Alerts use action groups, which are collections of actions that are triggered by the alert. Runbooks are just one of the many actions that you can use with action groups.

1. In your Automation account, select **Alerts** under **Monitoring**.
2. Select **+ New alert rule**.
3. Click **Select** under **Resource**. On the **Select a resource** page, select your VM to alert off of, and click **Done**.
4. Click **Add condition** under **Condition**. Select the signal you want to use, for example **Percentage CPU** and click **Done**.
5. On the **Configure signal logic** page, enter your **Threshold value** under **Alert logic**, and click **Done**.
6. Under **Action groups**, select **Create New**.
7. On the **Add action group** page, give your action group a name and a short name.
8. Give the action a name. For the action type, select **Automation Runbook**.
9. Select **Edit Details**. On the **Configure Runbook** page, under **Runbook source**, select **User**.
10. Select your **Subscription** and **Automation account**, and then select the **Stop-AzureVmInResponsetoVMAalert** runbook.
11. Select **Yes** for **Enable the common alert schema**.
12. To create the action group, select **OK**.

The screenshot shows two overlapping dialog boxes from the Azure portal:

- Add action group** (Left):
 - Action group name: SampleCPUAlert
 - Short name: SampleCPU
 - Subscription: Contoso Subscription
 - Resource group: Default-ActivityLogAlerts
 - Actions** section:

ACTION NAME	ACTION TYPE	STATUS	DETAILS	ACTIONS
Runbook	Automation Runbook	Enabled		Edit details (button)

Please configure the action by clicking the link.
- Configure Runbook** (Right):
 - Run runbook: Enabled (selected)
 - Runbook source: Built-in (selected)
 - Subscription: Contoso Subscription
 - Automation account: ExampleAutomation
 - Runbook: Stop-AzureVmInResponseToVMAalert
 - Parameters** section:

Configure parameters	>
----------------------	---

Info When the alert is triggered, the alert data will be passed to the runbook in the \$WebhookData input parameter. [See this article for more information.](#)
 - Enable the common alert schema** (Yes selected)
 - Warning** If this action is a user runbook and you are changing the schema type then you may need to adjust the user runbook to handle the new schema type. If this action is a built-in runbook then you can choose any schema type and the built-in runbook is already able to process it.

You can use this action group in the [activity log alerts](#) and [near real-time alerts](#) that you create.

13. Under **Alert Details**, add an alert rule name and description and click **Create alert rule**.

Next steps

- To start a runbook using a webhook, see [Start a runbook from a webhook](#).
- To discover different ways to start a runbook, see [Start a runbook](#).
- To create an activity log alert, see [Create activity log alerts](#).
- To learn how to create a near real-time alert, see [Create an alert rule in the Azure portal](#).
- For a PowerShell cmdlet reference, see [Az.Automation](#).

Track updated files with a watcher task

9/10/2021 • 5 minutes to read • [Edit Online](#)

Azure Automation uses a watcher task to look for events and trigger actions with PowerShell runbooks. The watcher task contains two parts, the watcher and the action. A watcher runbook runs at an interval defined in the watcher task, and outputs data to an action runbook.

NOTE

Watcher tasks are not supported in Azure China Vianet 21.

IMPORTANT

Starting in May 2020, using Azure Logic Apps is the recommended and supported way to monitor for events, schedule recurring tasks, and trigger actions. See [Schedule and run recurring automated tasks, processes, and workflows with Azure Logic Apps](#).

This article walks you through creating a watcher task to monitor when a new file is added to a directory. You learn how to:

- Import a watcher runbook
- Create an Automation variable
- Create an action runbook
- Create a watcher task
- Trigger a watcher
- Inspect the output

Prerequisites

To complete this article, the following are required:

- Azure subscription. If you don't have one yet, you can [activate your MSDN subscriber benefits](#) or sign up for a [free account](#).
- [Automation account](#) to hold the watcher and action runbooks and the Watcher Task.
- A [hybrid runbook worker](#) where the watcher task runs.
- PowerShell runbooks. PowerShell Workflow runbooks aren't supported by watcher tasks.

Import a watcher runbook

This article uses a watcher runbook called **Watcher runbook that looks for new files in a directory** to look for new files in a directory. The watcher runbook retrieves the last known write time to the files in a folder and looks at any files newer than that watermark.

You can import this runbook into your Automation account from the portal using the following steps.

1. Sign in to the [Azure portal](#).
2. Search for and select **Automation Accounts**.
3. On the **Automation Accounts** page, select the name of your Automation account from the list.
4. In the left pane, select **Runbooks gallery** under **Process Automation**.

5. Make sure **GitHub** is selected in the **Source** drop-down list.
6. Search for **Watcher runbook**.
7. Select **Watcher runbook** that looks for new files in a directory, and select **Import** on the details page.
8. Give the runbook a name and optionally a description and click **OK** to import the runbook into your Automation account. You should see an **Import successful** message in a pane at the upper right of your window.
9. The imported runbook appears in the list under the name you gave it when you select Runbooks from the left-hand pane.
10. Click on the runbook, and on the runbook details page, select **Edit** and then click **Publish**. When prompted, click **Yes** to publish the runbook.

You can also download the runbook from the [Azure Automation GitHub organization](#).

1. Navigate to the Azure Automation GitHub organization page for [Watch-NewFile.ps1](#).
2. To download the runbook from GitHub, select **Code** from the right-hand side of the page, and then select **Download ZIP** to download the whole code in a zip file.
3. Extract the contents and [import the runbook](#).

Create an Automation variable

An [Automation variable](#) is used to store the timestamps that the preceding runbook reads and stores from each file.

1. Select **Variables** under Shared Resources and click **+ Add a variable**.
2. Enter **Watch-NewFileTimestamp** for the name.
3. Select **DateTime** for the type. It will default to the current date and time.

New Variable

Name *

Description

Type ⓘ

DateTime

Value (Local time) * ⓘ

04/29/2021 3:16:50 PM

Encrypted *

Yes No

Create

4. Click **Create** to create the Automation variable.

Create an action runbook

An action runbook is used in a watcher task to act on the data passed to it from a watcher runbook. You must import a predefined action runbook, either from the Azure portal or from the [Azure Automation GitHub organization](#).

You can import this runbook into your Automation account from the Azure portal:

1. Sign in to the [Azure portal](#).
2. Search for and select **Automation Accounts**.
3. On the **Automation Accounts** page, select the name of your Automation account from the list.
4. In the left pane, select **Runbooks gallery** under **Process Automation**.
5. Make sure **GitHub** is selected in the **Source** drop-down list.
6. Search for **Watcher action**, select **Watcher action that processes events triggered by a watcher runbook**, and click **Import**.
7. Optionally, change the name of the runbook on the import page, and then click **OK** to import the runbook.
You should see an **Import successful** message in the notification pane in the upper right-hand side of the browser.
8. Go to your Automation Account page, and click on **Runbooks** on the left. Your new runbook should be listed under the name you gave it in the previous step. Click on the runbook, and on the runbook details page, select **Edit** and then click **Publish**. When prompted, click **Yes** to publish the runbook.

To create an action runbook by downloading it from the [Azure Automation GitHub organization](#):

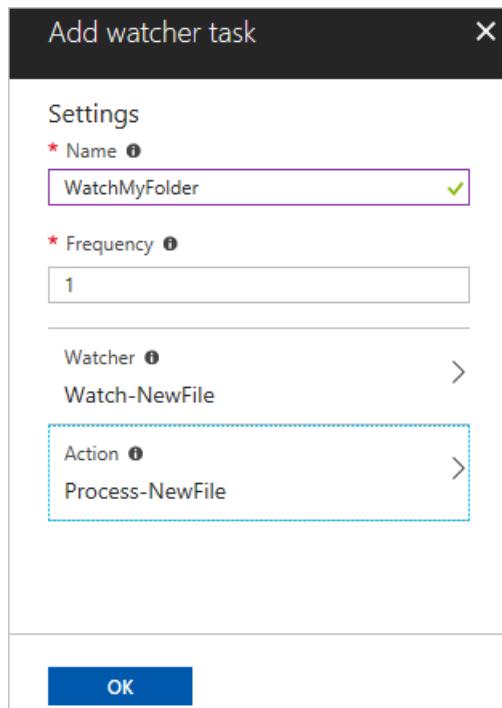
1. Navigate to the Azure Automation GitHub organization page for [Process-NewFile.ps1](#).
2. To download the runbook from GitHub, select **Code** from the right-hand side of the page, and then select **Download ZIP** to download the whole code in a zip file.
3. Extract the contents and [import the runbook](#).

Create a watcher task

In this step, you configure the watcher task referencing the watcher and action runbooks defined in the preceding sections.

1. Navigate to your Automation account and select **Watcher tasks** under **Process Automation**.
2. Select the Watcher tasks page and click **+ Add a watcher task**.
3. Enter **WatchMyFolder** as the name.
4. Select **Configure watcher** and choose the **Watch-NewFile** runbook.
5. Enter the following values for the parameters:
 - **FOLDERPATH** - A folder on the Hybrid Runbook Worker where new files get created, for example, `d:\examplefiles`.
 - **EXTENSION** - Extension for the configuration. Leave blank to process all file extensions.
 - **RECURSE** - Recursive operation. Leave this value as the default.
 - **RUN SETTINGS** - Setting for running the runbook. Pick the hybrid worker.
6. Click **OK**, and then **Select** to return to the Watcher page.
7. Select **Configure action** and choose the **Process-NewFile** runbook.
8. Enter the following values for the parameters:

- **EVENTDATA** - Event data. Leave blank. Data is passed in from the watcher runbook.
 - **Run Settings** - Setting for running the runbook. Leave as Azure, as this runbook runs in Azure Automation.
9. Click **OK**, and then **Select** to return to the Watcher page.
10. Click **OK** to create the watcher task.



Trigger a watcher

You must run a test as described below to ensure that the watcher task works as expected.

1. Remote into the Hybrid Runbook Worker.
2. Open **Powershell** and create a test file in the folder.

```
New-Item -Name ExampleFile1.txt
```

The following example shows the expected output.

Directory: D:\examplefiles		
Mode	LastWriteTime	Length Name
----	-----	-----
-a---	12/11/2017 9:05 PM	0 ExampleFile1.txt

Inspect the output

1. Navigate to your Automation account and select **Watcher tasks** under **Process Automation**.
2. Select the watcher task **WatchMyFolder**.
3. Click on **View watcher streams** under **Streams** to see that the watcher has found the new file and started the action runbook.
4. To see the action runbook jobs, click on **View watcher action jobs**. Each job can be selected to view the

details of the job.

The screenshot shows two windows side-by-side. The left window is titled 'WatchMyFolder' and contains the configuration for a 'Watcher task'. It includes sections for 'Settings' (Name: WatchMyFolder, Last modified: 11/11/2017 8:58 PM, Frequency: 1 minute), 'Watcher' (Runbook: Watch-NewFile, Streams: View watcher streams), and 'Action' (Runbook: Process-NewFile, Jobs: View watcher action jobs). The right window is titled 'Jobs' and lists completed tasks with their status, creation date, and last update date. The table has columns for STATUS, CREATED, and LAST UPDATED.

STATUS	CREATED	LAST UPDATED
✓ Completed	11/11/2017 1:13 PM	11/11/2017 1:13 PM
✓ Completed	11/11/2017 1:11 PM	11/11/2017 1:11 PM
✓ Completed	11/11/2017 1:09 PM	11/11/2017 1:09 PM
✓ Completed	11/11/2017 12:59 PM	11/11/2017 12:59 PM

The expected output when the new file is found can be seen in the following example:

```
Message is Process new file...
Passed in data is @{FileName=D:\examplefiles\ExampleFile1.txt; Length=0}
```

Next steps

To learn more about authoring your own runbook, see [Create a PowerShell runbook](#).

Manage Office 365 services

9/10/2021 • 4 minutes to read • [Edit Online](#)

You can use Azure Automation for management of Office 365 subscription services, for products such as Microsoft Word and Microsoft Outlook. Interactions with Office 365 are enabled by [Azure Active Directory \(Azure AD\)](#). See [Use Azure AD in Azure Automation to authenticate to Azure](#).

Prerequisites

You need the following to manage Office 365 subscription services in Azure Automation.

- An Azure subscription. See [Subscription decision guide](#).
- An Automation object in Azure to hold the user account credentials and runbooks. See [An introduction to Azure Automation](#).
- Azure AD. See [Use Azure AD in Azure Automation to authenticate to Azure](#).
- An Office 365 tenant, with an account. See [Set up your Office 365 tenant](#).

Install the MSOnline and MSOnlineExt modules

Use of Office 365 within Azure Automation requires Microsoft Azure Active Directory for Windows PowerShell (`MSOnline` module). You'll also need the module `MSOnlineExt`, which simplifies Azure AD management in single- and multi-tenant environments. Install the modules as described in [Use Azure AD in Azure Automation to authenticate to Azure](#).

NOTE

To use MSOnline PowerShell, you must be a member of Azure AD. Guest users can't use the module.

Create an Azure Automation account

To complete the steps in this article, you need an account in Azure Automation. See [Create an Azure Automation account](#).

Add MSOnline and MSOnlineExt as assets

Now add the installed MSOnline and MSOnlineExt modules to enable Office 365 functionality. Refer to [Manage modules in Azure Automation](#).

1. In the Azure portal, select **Automation Accounts**.
2. Choose your Automation account.
3. Select **Modules Gallery** under **Shared Resources**.
4. Search for MSOnline.
5. Select the `MSOnline` PowerShell module and click **Import** to import the module as an asset.
6. Repeat steps 4 and 5 to locate and import the `MSOnlineExt` module.

Create a credential asset (optional)

It's optional to create a credential asset for the Office 365 administrative user who has permissions to run your script. It can help, though, to keep from exposing user names and passwords inside PowerShell scripts. For

instructions, see [Create a credential asset](#).

Create an Office 365 service account

To run Office 365 subscription services, you need an Office 365 service account with permissions to do what you want. You can use one global administrator account, one account per service, or have one function or script to execute. In any case, the service account requires a complex and secure password. See [Set up Office 365 for business](#).

Connect to the Azure AD online service

NOTE

To use the MSOnline module cmdlets, you must run them from Windows PowerShell. PowerShell Core does not support these cmdlets.

You can use the MSOnline module to connect to Azure AD from the Office 365 subscription. The connection uses an Office 365 user name and password or uses multi-factor authentication (MFA). You can connect using the Azure portal or a Windows PowerShell command prompt (does not have to be elevated).

A PowerShell example is shown below. The `Get-Credential` cmdlet prompts for credentials and stores them in the `$Msolcred` variable. Then the `Connect-MsolService` cmdlet uses the credentials to connect to the Azure directory online service. If you want to connect to a specific Azure environment, use the `AzureEnvironment` parameter.

```
$Msolcred = Get-Credential  
Connect-MsolService -Credential $Msolcred -AzureEnvironment "AzureCloud"
```

If you don't receive any errors, you've connected successfully. A quick test is to run an Office 365 cmdlet, for example, `Get-MsolUser`, and see the results. If you receive errors, note that a common problem is an incorrect password.

NOTE

You can also use the AzureRM module or the Az module to connect to Azure AD from the Office 365 subscription. The main connection cmdlet is `Connect-AzureAD`. This cmdlet supports the `AzureEnvironmentName` parameter for specific Office 365 environments.

Create a PowerShell runbook from an existing script

You access Office 365 functionality from a PowerShell script. Here's an example of a script for a credential named `Office-Credentials` with user name of `admin@TenantOne.com`. It uses `Get-AutomationPSCredential` to import the Office 365 credential.

```

$emailFromAddress = "admin@TenantOne.com"
$emailToAddress = "servicedesk@TenantOne.com"
$emailSMTPServer = "outlook.office365.com"
$emailSubject = "Office 365 License Report"

$credObject = Get-AutomationPSCredential -Name "Office-Credentials"
Connect-MsolService -Credential $credObject

$O365Licenses = Get-MsolAccountSku | Out-String
Send-MailMessage -Credential $credObject -From $emailFromAddress -To $emailToAddress -Subject $emailSubject
-Body $O365Licenses -SmtpServer $emailSMTPServer -UseSSL

```

Run the script in a runbook

You can use your script in an Azure Automation runbook. For example purposes, we'll use the PowerShell runbook type.

1. Create a new PowerShell runbook. Refer to [Create an Azure Automation runbook](#).
2. From your Automation account, select **Runbooks** under **Process Automation**.
3. Select the new runbook and click **Edit**.
4. Copy your script and paste it into the textual editor for the runbook.
5. Select **ASSETS**, then expand **Credentials** and verify that the Office 365 credential is there.
6. Click **Save**.
7. Select **Test pane**, then click **Start** to begin testing your runbook. See [Manage runbooks in Azure Automation](#).
8. When testing is complete, exit from the Test pane.

Publish and schedule the runbook

To publish and then schedule your runbook, see [Manage runbooks in Azure Automation](#).

Next steps

- For details of credential use, see [Manage credentials in Azure Automation](#).
- For information about modules, see [Manage modules in Azure Automation](#).
- If you need to start a runbook, see [Start a runbook in Azure Automation](#).
- For PowerShell details, see [PowerShell Docs](#).

Deploy an Amazon Web Services VM with a runbook

4/2/2021 • 4 minutes to read • [Edit Online](#)

In this article, you learn how you can leverage Azure Automation to provision a virtual machine in your Amazon Web Service (AWS) subscription and give that VM a specific name - which AWS refers to as "tagging" the VM.

Prerequisites

You need to have an Azure Automation account and an Amazon Web Services (AWS) subscription. For more information on setting up an Azure Automation account and configuring it with your AWS subscription credentials, review [Configure Authentication with Amazon Web Services](#). This account should be created or updated with your AWS subscription credentials before proceeding, as you reference this account in the sections below.

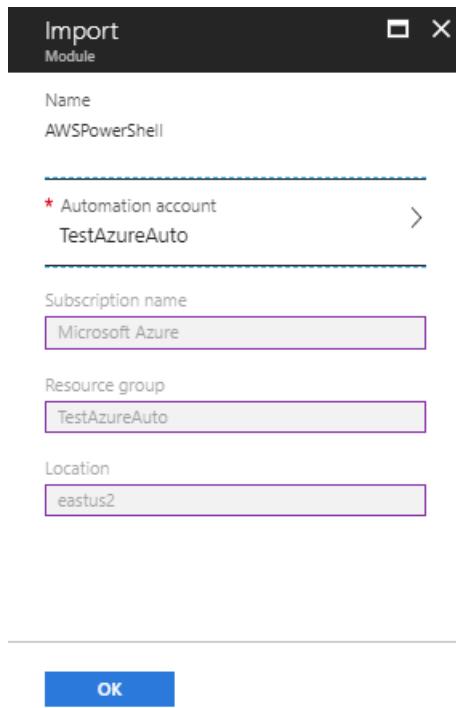
Deploy Amazon Web Services PowerShell Module

Your VM provisioning runbook uses the AWS PowerShell module to do its work. Use the following steps to add the module to your Automation account that is configured with your AWS subscription credentials.

1. Open your web browser and navigate to the [PowerShell Gallery](#) and click on the **Deploy to Azure Automation** button.

The screenshot shows the PowerShell Gallery interface. At the top, there's a navigation bar with links for Home, Get Started, Modules (which is underlined), Scripts, Publish, and Statistics. There's also a search bar labeled 'Search Items' with a magnifying glass icon. On the right side of the header, there are 'Register | Sign in' links. The main content area displays a module named 'AWS Tools for Windows PowerShell 3.1.58.0'. It features a large blue PowerShell icon. Below the icon, the download count '1,119 Downloads' and the last publish date '2016-04-07 Last published' are shown. To the right of these stats, there are three buttons: 'Inspect' (with a PS command example), 'Install' (with a PS command example), and 'Deploy' (with a 'Deploy to Azure Automation' button). The 'Deploy' button has a gear icon and is highlighted with a blue border.

2. You are taken to the Azure login page and after authenticating, you will be routed to the Azure portal and presented with the following page:



3. Select the Automation account to use and click OK to start deployment.

NOTE

When Azure Automation imports a PowerShell module, it extracts the cmdlets. The activities don't appear until Automation has completely finished importing the module and extracting the cmdlets. This process can take a few minutes.

4. In the Azure portal, open your Automation account.
5. Click on the **Assets** tile and, on the Assets pane, select **Modules**.
6. On the Modules page, you see the **AWSPowerShell** module in the list.

Create AWS deploy VM runbook

Once the AWS PowerShell Module has been deployed, you can now author a runbook to automate provisioning a virtual machine in AWS using a PowerShell script. The steps below demonstrate how to use native PowerShell script in Azure Automation.

NOTE

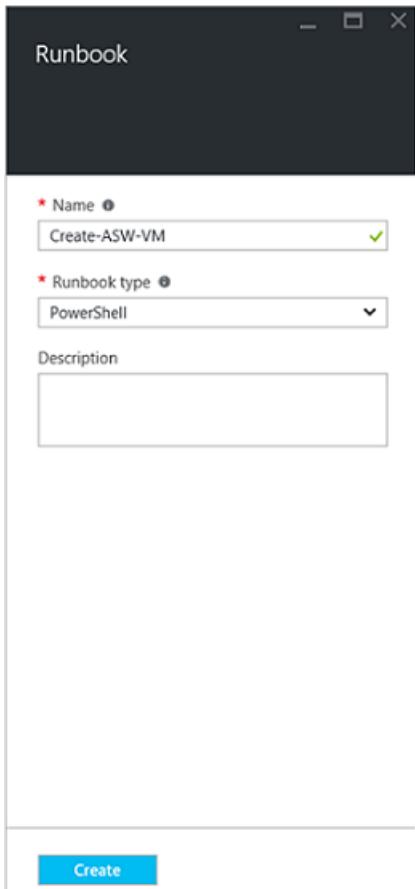
For further options and information regarding this script, please visit the [PowerShell Gallery](#).

1. Download the PowerShell script New-AwsVM from the PowerShell Gallery by opening a PowerShell session and typing the following command:

```
Save-Script -Name New-AwsVM -Path <path>
```

2. From the Azure portal, open your Automation account and select **Runbooks** under **Process Automation**.

3. From the Runbooks page, select **Add a runbook**.
4. On the Add a runbook pane, select **Quick Create** to create a new runbook.
5. On the Runbook properties pane, type in a name for your runbook.
6. From the **Runbook type** drop-down list, select **PowerShell**, and then click **Create**.



7. When the Edit PowerShell Runbook page appears, copy and paste the PowerShell script into the runbook authoring canvas.

```

34 #>
35 #ToDo:
36 #Change the default values for the following parameters if they do not match with yours:
37 #AWSRegion, EC2ImageName, MinCount, MaxCount, InstanceType
38 #Create an Azure Automation Asset called "AwsCred"
39 #Turn on Log verbose records and optionally Log progress records under the runbook settings to see verbose
40
41 param (
42     [Parameter(Mandatory=$true)]
43     [string]$VmName,
44     [ValidateNotNullOrEmpty()]
45     [string]$AWSRegion = "us-west-2",
46     [ValidateNotNullOrEmpty()]
47     [string]$EC2ImageName = "WINDOWS_2012R2_BASE",
48     [ValidateNotNullOrEmpty()]
49     [string]$MinCount = 1,
50     [ValidateNotNullOrEmpty()]
51     [string]$MaxCount = 1,
52     [ValidateNotNullOrEmpty()]
53     [string]$InstanceType = "t2.micro"
54 )
55
56 # Get credentials to authenticate against AWS
57 $AwsCred = Get-AutomationPSCredential -Name "AwsCred"
58 $AwsAccessKeyId = $AwsCred.UserName
59 $AwsSecretKey = $AwsCred.GetNetworkCredential().Password

```

Note the following when working with the example PowerShell script:

- The runbook contains a number of default parameter values. Evaluate all default values and update where necessary.

- If you have stored your AWS credentials as a credential asset named differently from `AWScred`, you need to update the script on line 57 to match accordingly.
 - When working with the AWS CLI commands in PowerShell, especially with this example runbook, you must specify the AWS region. Otherwise, the cmdlets fail. View AWS topic [Specify AWS Region](#) in the AWS Tools for PowerShell document for further details.
8. To retrieve a list of image names from your AWS subscription, launch PowerShell ISE and import the AWS PowerShell Module. Authenticate against AWS by replacing `Get-AutomationPSCredential` in your ISE environment with `AWScred = Get-Credential`. This statement prompts for your credentials and you can provide your access key ID for the user name and your secret access key for the password.

```
#Sample to get the AWS VM available images
#Please provide the path where you have downloaded the AWS PowerShell module
Import-Module AWSPowerShell
$AwsRegion = "us-west-2"
$AwsCred = Get-Credential
$AwsAccessKeyId = $AwsCred.UserName
$AwsSecretKey = $AwsCred.GetNetworkCredential().Password

# Set up the environment to access AWS
Set-AwsCredentials -AccessKey $AwsAccessKeyId -SecretKey $AwsSecretKey -StoreAs AWSProfile
Set-DefaultAWSRegion -Region $AwsRegion

Get-EC2ImageByName -ProfileName AWSProfile
```

The following output is returned:

```
PS C:\> Get-EC2ImageByName -ProfileName AWSProfile
WINDOWS_2012R2_BASE
WINDOWS_2012R2_SQL_SERVER_EXPRESS_2014
WINDOWS_2012R2_SQL_SERVER_STANDARD_2014
WINDOWS_2012R2_SQL_SERVER_WEB_2014
WINDOWS_2012_BASE
WINDOWS_2012_SQL_SERVER_EXPRESS_2014
WINDOWS_2012_SQL_SERVER_STANDARD_2014
WINDOWS_2012_SQL_SERVER_WEB_2014
WINDOWS_2012_SQL_SERVER_EXPRESS_2012
WINDOWS_2012_SQL_SERVER_STANDARD_2012
WINDOWS_2012_SQL_SERVER_WEB_2012
WINDOWS_2012_SQL_SERVER_EXPRESS_2008
WINDOWS_2012_SQL_SERVER_STANDARD_2008
WINDOWS_2012_SQL_SERVER_WEB_2008
WINDOWS_2008R2_BASE
WINDOWS_2008R2_SQL_SERVER_EXPRESS_2012
WINDOWS_2008R2_SQL_SERVER_STANDARD_2012
WINDOWS_2008R2_SQL_SERVER_WEB_2012
WINDOWS_2008R2_SQL_SERVER_EXPRESS_2008
WINDOWS_2008R2_SQL_SERVER_STANDARD_2008
WINDOWS_2008R2_SQL_SERVER_WEB_2008
WINDOWS_2008RTM_BASE
WINDOWS_2008RTM_SQL_SERVER_EXPRESS_2008
WINDOWS_2008RTM_SQL_SERVER_STANDARD_2008
WINDOWS_2008_BEANSTALK_IIS75
WINDOWS_2012_BEANSTALK_IIS8
VPC_NAT
```

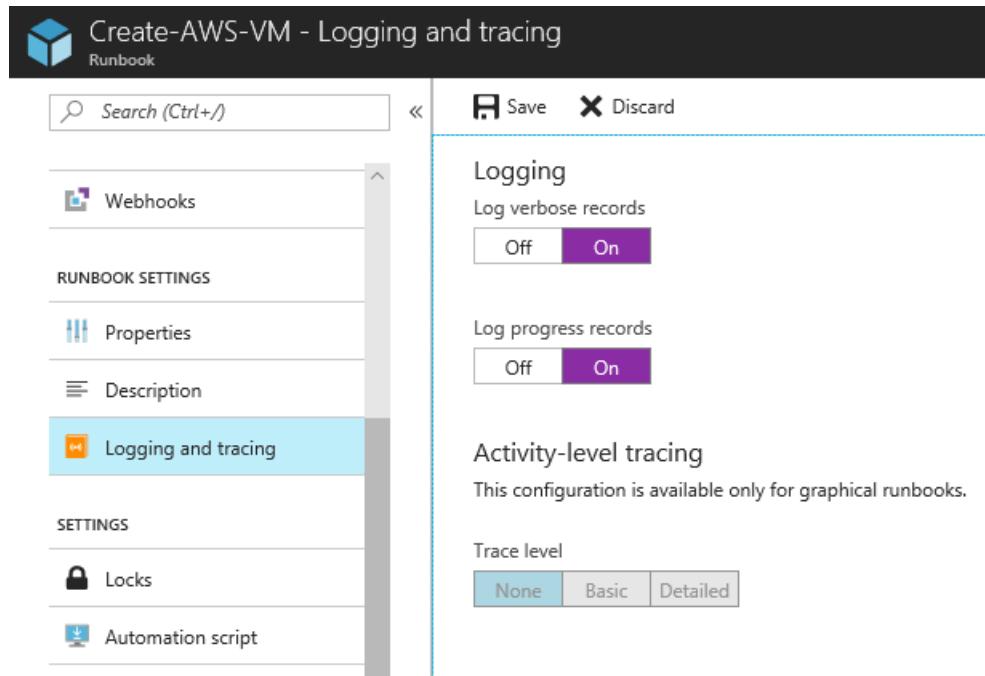
9. Copy and paste the one of the image names in an Automation variable as referenced in the runbook as `$InstanceType`. Since, in this example, you are using the free AWS tiered subscription, you use `t2.micro` for your runbook example.
10. Save the runbook, then click **Publish** to publish the runbook and then **Yes** when prompted.

Test the AWS VM runbook

1. Verify that the runbook creates an asset called `AWScred` for authenticating against AWS, or update the script to reference your credential asset name.
2. Verify your new runbook and make sure that all parameter values have been updated has necessary. Ensure

that the AWS PowerShell module has been imported into Azure Automation.

3. In Azure Automation, set **Log verbose records** and optionally **Log progress records** under the runbook operation **Logging and tracing** to On.



4. Click **Start** to start the runbook, then click **OK** when the Start Runbook pane opens.
5. On the Start Runbook pane, provide a VM name. Accept the default values for the other parameters that you preconfigured in the script. Click **OK** to start the runbook job.



Parameters

* VMNAME ⓘ

Enter a value

Mandatory, String

AWSREGION ⓘ

Default will be used

Optional, String, Default: "us-west-2"

EC2IMAGENAME ⓘ

Default will be used

Optional, String, Default: "WINDOWS_2012R2_BASE"

MINCOUNT ⓘ

Default will be used

Optional, String, Default: 1

MAXCOUNT ⓘ

Default will be used

Optional, String, Default: 1

INSTANCETYPE ⓘ

Default will be used

Optional, String, Default: "t2.micro"

Run Settings

Run on ⓘ

Azure Hybrid Worker

OK

6. A Job pane is opened for the runbook job that you created. Close this pane.

7. You can view progress of the job and view output streams by selecting All Logs from the runbook Job pane.

A screenshot of the Azure Runbook Job pane for a job named "New-AwsVM 1/7/2016, 1:29 PM". The pane has tabs for "Job", "Streams", and "Logs". The "Streams" tab is highlighted with a red box. On the left, there's an "Overview" section with a "Job Summary" table and a "Status" section showing 0 errors and 0 warnings. The "Logs" section on the right shows a table of log entries with columns for TIME, TYPE, and DETAILS. The logs show the progression of creating a new AWS VM, including preparing modules, authenticating, loading credentials, getting the AWS image, creating the instance, applying the new VM name, and successfully creating the VM named "MyAwsVM".

TIME	TYPE	DETAILS
1/7/2016, 1:30 PM	→ Progress	Preparing modules for first use.
1/7/2016, 1:30 PM	Verbose	Authenticating against AWS...
1/7/2016, 1:30 PM	→ Progress	Preparing modules for first use.
1/7/2016, 1:30 PM	Verbose	Credentials loaded from the supplied key parameters.
1/7/2016, 1:30 PM	Verbose	Getting AWS image...
1/7/2016, 1:30 PM	Verbose	The following image has been found: Windows_Server-2012-R2-RTM...
1/7/2016, 1:30 PM	Verbose	Creating new AWS Instance...
1/7/2016, 1:30 PM	Output	
1/7/2016, 1:30 PM	Verbose	Applying new VM Name...
1/7/2016, 1:30 PM	Verbose	Successfully created AWS VM: MyAwsVM

8. To confirm that the VM is being provisioned, log into the AWS Management Console if you aren't currently logged in.

Filter by tags and attributes or search by keyword						
	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks
	Hoeba1	i-4439b59d	m1.small	us-west-2a	terminated	
	MyAWSVM1	i-66ce41bf	m1.small	us-west-2a	running	Initializing
	AWS-RH1	i-a89e706c	t2.micro	us-west-2b	running	2/2 checks...

Next steps

- To find out what runbooks are supported, see [Azure Automation runbook types](#).
- To work with runbooks, see [Manage runbooks in Azure Automation](#).
- For details of PowerShell, see [PowerShell Docs](#).
- For script support, see [Native PowerShell script support in Azure Automation](#).
- For a PowerShell cmdlet reference, see [Az.Automation](#).

Deploy an Azure Resource Manager template in a PowerShell runbook

9/10/2021 • 4 minutes to read • [Edit Online](#)

You can write an [Azure Automation PowerShell runbook](#) that deploys an Azure resource by using an [Azure Resource Manager template](#). The templates allow you to use Azure Automation to automate deployment of your Azure resources. You can maintain your Resource Manager templates in a central, secure location, such as Azure Storage.

In this article, we create a PowerShell runbook that uses a Resource Manager template stored in [Azure Storage](#) to deploy a new Azure Storage account.

Prerequisites

- Azure subscription. If you don't have one yet, you can [activate your MSDN subscriber benefits](#) or [sign up for a free account](#).
- [Automation account](#) to hold the runbook and authenticate to Azure resources. This account must have permission to start and stop the virtual machine.
- [Azure Storage account](#) in which to store the Resource Manager template.
- Azure PowerShell installed on a local machine. See [Install the Azure PowerShell Module](#) for information about how to get Azure PowerShell.

Create the Resource Manager template

In this example, we use a Resource Manager template that deploys a new Azure Storage account.

In a text editor, copy the following text:

```
{
    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
        "storageAccountType": {
            "type": "string",
            "defaultValue": "Standard_LRS",
            "allowedValues": [
                "Standard_LRS",
                "Standard_GRS",
                "Standard_ZRS",
                "Premium_LRS"
            ],
            "metadata": {
                "description": "Storage Account type"
            }
        },
        "location": {
            "type": "string",
            "defaultValue": "[resourceGroup().location]",
            "metadata": {
                "description": "Location for all resources."
            }
        }
    },
    "variables": {
        "storageAccountName": "[concat(uniquestring(resourceGroup().id), 'standardsa')]"
    },
    "resources": [
        {
            "type": "Microsoft.Storage/storageAccounts",
            "name": "[variables('storageAccountName')]",
            "apiVersion": "2018-02-01",
            "location": "[parameters('location')]",
            "sku": {
                "name": "[parameters('storageAccountType')]"
            },
            "kind": "Storage",
            "properties": {}
        }
    ],
    "outputs": {
        "storageAccountName": {
            "type": "string",
            "value": "[variables('storageAccountName')]"
        }
    }
}
```

Save the file locally as **TemplateTest.json**.

Save the Resource Manager template in Azure Files

Now we use PowerShell to create an Azure file share and upload the **TemplateTest.json** file. For instructions on how to create a file share and upload a file in the Azure portal, see [Get started with Azure Files on Windows](#).

Launch PowerShell on your local machine, and run the following commands to create a file share and upload the Resource Manager template to that file share.

```
# Log into Azure
Connect-AzAccount

# Get the access key for your storage account
$key = Get-AzStorageAccountKey -ResourceGroupName 'MyAzureAccount' -Name 'MyStorageAccount'

# Create an Azure Storage context using the first access key
$context = New-AzStorageContext -StorageAccountName 'MyStorageAccount' -StorageAccountKey $key[0].value

# Create a file share named 'resource-templates' in your Azure Storage account
$fileShare = New-AzStorageShare -Name 'resource-templates' -Context $context

# Add the TemplateTest.json file to the new file share
# "TemplatePath" is the path where you saved the TemplateTest.json file
$templateFile = 'C:\TemplatePath'
Set-AzStorageFileContent -ShareName $fileShare.Name -Context $context -Source $templateFile
```

Create the PowerShell runbook script

Now we create a PowerShell script that gets the `TemplateTest.json` file from Azure Storage and deploy the template to create a new Azure Storage account.

In a text editor, paste the following text:

```

param (
    [Parameter(Mandatory=$true)]
    [string]
    $ResourceGroupName,

    [Parameter(Mandatory=$true)]
    [string]
    $StorageAccountName,

    [Parameter(Mandatory=$true)]
    [string]
    $StorageAccountKey,

    [Parameter(Mandatory=$true)]
    [string]
    $StorageFileName
)

# Authenticate to Azure if running from Azure Automation
$ServicePrincipalConnection = Get-AutomationConnection -Name "AzureRunAsConnection"
Connect-AzAccount `-
    -ServicePrincipal `-
    -Tenant $ServicePrincipalConnection.TenantId `-
    -ApplicationId $ServicePrincipalConnection.ApplicationId `-
    -CertificateThumbprint $ServicePrincipalConnection.CertificateThumbprint | Write-Verbose

#Set the parameter values for the Resource Manager template
$Parameters = @{
    "storageAccountType"="Standard_LRS"
}

# Create a new context
$Context = New-AzStorageContext -StorageAccountName $StorageAccountName -StorageAccountKey
$StorageAccountKey

Get-AzStorageFileContent -ShareName 'resource-templates' -Context $Context -path 'TemplateTest.json' - 
Destination 'C:\Temp'

$TemplateFile = Join-Path -Path 'C:\Temp' -ChildPath $StorageFileName

# Deploy the storage account
New-AzResourceGroupDeployment -ResourceGroupName $ResourceGroupName -TemplateFile $TemplateFile - 
TemplateParameterObject $Parameters

```

Save the file locally as **DeployTemplate.ps1**.

Import and publish the runbook into your Azure Automation account

Now we use PowerShell to import the runbook into your Azure Automation account, and then publish the runbook. For information about how to import and publish a runbook in the Azure portal, see [Manage runbooks in Azure Automation](#).

To import **DeployTemplate.ps1** into your Automation account as a PowerShell runbook, run the following PowerShell commands:

```

# MyPath is the path where you saved DeployTemplate.ps1
# MyResourceGroup is the name of the Azure ResourceGroup that contains your Azure Automation account
# MyAutomationAccount is the name of your Automation account
$importParams = @{
    Path = 'C:\MyPath\DeployTemplate.ps1'
    ResourceGroupName = 'MyResourceGroup'
    AutomationAccountName = 'MyAutomationAccount'
    Type = 'PowerShell'
}
Import-AzAutomationRunbook @importParams

# Publish the runbook
$publishParams = @{
    ResourceGroupName = 'MyResourceGroup'
    AutomationAccountName = 'MyAutomationAccount'
    Name = 'DeployTemplate'
}
Publish-AzAutomationRunbook @publishParams

```

Start the runbook

Now we start the runbook by calling the [Start-AzAutomationRunbook](#) cmdlet. For information about how to start a runbook in the Azure portal, see [Starting a runbook in Azure Automation](#).

Run the following commands in the PowerShell console:

```

# Set up the parameters for the runbook
$runbookParams = @{
    ResourceGroupName = 'MyResourceGroup'
    StorageAccountName = 'MyStorageAccount'
    StorageAccountKey = $key[0].Value # We got this key earlier
    StorageFileName = 'TemplateTest.json'
}

# Set up parameters for the Start-AzAutomationRunbook cmdlet
$startParams = @{
    ResourceGroupName = 'MyResourceGroup'
    AutomationAccountName = 'MyAutomationAccount'
    Name = 'DeployTemplate'
    Parameters = $runbookParams
}

# Start the runbook
$job = Start-AzAutomationRunbook @startParams

```

After the runbook runs, you can check its status by retrieving the property value of the job object `$job.Status`.

The runbook gets the Resource Manager template and uses it to deploy a new Azure Storage account. You can see the new storage account was created by running the following command:

```
Get-AzStorageAccount
```

Next steps

- To learn more about Resource Manager templates, see [Azure Resource Manager overview](#).
- To get started with Azure Storage, see [Introduction to Azure Storage](#).
- To find other useful Azure Automation runbooks, see [Use runbooks and modules in Azure Automation](#).
- For a PowerShell cmdlet reference, see [Az.Automation](#).

Tutorial: Integrate Azure Automation with Event Grid and Microsoft Teams

11/2/2020 • 3 minutes to read • [Edit Online](#)

In this tutorial, you learn how to:

- Import an Event Grid sample runbook.
- Create an optional Microsoft Teams webhook.
- Create a webhook for the runbook.
- Create an Event Grid subscription.
- Create a VM that triggers the runbook.

If you don't have an Azure subscription, create a [free account](#) before you begin.

Prerequisites

IMPORTANT

Using this Azure feature from PowerShell requires the `AzureRM` module installed. This is an older module only available for Windows PowerShell 5.1 that no longer receives new features. The `Az` and `AzureRM` modules are **not** compatible when installed for the same versions of PowerShell. If you need both versions:

1. [Uninstall the Az module](#) from a PowerShell 5.1 session.
2. [Install the AzureRM module](#) from a PowerShell 5.1 session.
3. [Download and install PowerShell Core 6.x or later](#).
4. [Install the Az module](#) in a PowerShell Core session.

To complete this tutorial, an [Azure Automation account](#) is required to hold the runbook that is triggered from the Azure Event Grid subscription.

- The `AzureRM.Tags` module needs to be loaded in your Automation Account, see [How to import modules in Azure Automation](#) to learn how to import modules into Azure Automation.

Import an Event Grid sample runbook

1. Select your Automation account, and select the [Runbooks](#) page.

NAME	AUTHORING STATUS	LAST MODIFIED	TAGS
AzureAutomationTutorial	✓ Published	8/14/2018, 9:27 AM	
AzureAutomationTutorialPython2	✓ Published	8/14/2018, 9:27 AM	
AzureAutomationTutorialScript	✓ Published	8/14/2018, 9:27 AM	
AzureClassicAutomationTutorial	✓ Published	8/14/2018, 9:27 AM	
AzureClassicAutomationTutorial...	✓ Published	8/14/2018, 9:27 AM	

2. Select the **Browse gallery** button.
3. Search for **Event Grid**, and select **Integrating Azure Automation with Event grid**.

Integrating Azure Automation with Event grid
PowerShell Runbook
This sample Automation runbook integrates with Azure event grid subscriptions to get notified when a write command is performed against an Azure VM. The runbook adds a cost tag to the VM if it doesn't exist. It also sends an optional notification to a Microsoft Tag: [Azure Automation](#), [Microsoft Azure Virtual Machines](#), [Event Grid](#)
Created by: SC Automation Product Team
417 downloads
Last updated: 11/28/2017

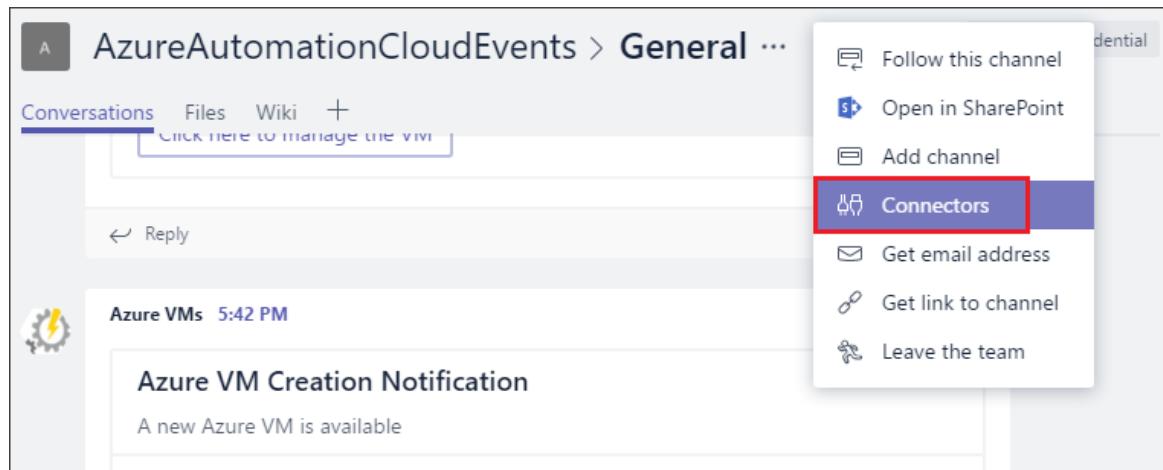
4. Select **Import** and name it **Watch-VMWrite**.
5. After it has imported, select **Edit** to view the runbook source.
6. Update the line 74 in the script to use `Tag` instead of `Tags`.

```
Update-AzureRmVM -ResourceGroupName $VMResourceGroup -VM $VM -Tag $Tag | Write-Verbose
```

7. Select the **Publish** button.

Create an optional Microsoft Teams webhook

1. In Microsoft Teams, select **More Options** next to the channel name, and then select **Connectors**.



2. Scroll through the list of connectors to **Incoming Webhook**, and select **Add**.
3. Enter **AzureAutomationIntegration** for the name, and select **Create**.
4. Copy the webhook URL to the clipboard, and save it. The webhook URL is used to send information to Microsoft Teams.
5. Select **Done** to save the webhook.

Create a webhook for the runbook

1. Open the Watch-VMWrite runbook.
2. Select **Webhooks**, and select the **Add Webhook** button.
3. Enter **WatchVMEventGrid** for the name. Copy the URL to the clipboard, and save it.

Two overlapping windows are shown. The left window is titled 'Add Webhook' and the right one is 'Create a new webhook'. Both windows have a header with a star, a close button, and a gear icon. The 'Create a new webhook' window is active. It contains a warning message: 'For security, after creating a webhook its URL can't be viewed. Make sure to copy it before pressing "OK", and to store it securely.' Below this are fields: 'Name' set to 'WatchVMEventGrid' with a green checkmark; 'Enabled' set to 'Yes'; 'Expires' set to '2018-12-06 10:12:36 AM'; and a 'URL' field containing 'https://s5events.azure-automation.net...' with a copy icon highlighted by a red box.

4. Select **Configure parameters and run settings**, and enter the Microsoft Teams webhook URL for **CHANNELURL**. Leave **WEBHOOKDATA** blank.

The screenshot shows the 'Add Webhook' configuration interface. On the left, under 'Webhook', the 'WatchVMEventGrid' runbook is selected. Under 'Parameters and run settings', 'Configure parameters and run settings' is chosen. On the right, the 'Parameters' tab is active, displaying two fields: 'WEBHOOKDATA' (with a note 'Optional, Object') and 'CHANNELURL' (containing the value 'https://outlook.office.com/webhook/52af4b29'). Below these is a 'Run Settings' section.

- Select **Create** to create the Automation runbook webhook.

Create an Event Grid subscription

- On the **Automation Account** overview page, select **Event grid**.

The screenshot shows the 'Event grid' blade within the 'ContosoFinance - Event grid' section of the Azure portal. The left sidebar lists 'Variables', 'RELATED RESOURCES' (including 'Linked workspace' and 'Event grid' which is highlighted with a red box), and 'ACCOUNT SETTINGS' (Properties, Keys, Pricing, Source control, Run as accounts). The main area features a blue circuit board icon and a message 'No Event Subscriptions Found! Consider modifying your search parameters above.' It includes a 'Topic Type' dropdown set to 'Event Hubs Namespaces', a 'Subscription' dropdown, and a 'Location' dropdown set to 'West Central US'. A 'Create one' button is visible at the bottom.

- Click **+ Event Subscription**.
- Configure the subscription with the following information:
 - For **Topic Type**, select **Azure Subscriptions**.
 - Uncheck the **Subscribe to all event types** check box.
 - Enter **AzureAutomation** for the name.
 - In the **Defined Event Types** drop-down, uncheck all options except **Resource Write Success**.

NOTE

Azure Resource Manager does not currently differentiate between Create and Update, so implementing this tutorial for all Microsoft.Resources.ResourceWriteSuccess events in your Azure Subscription could result in a high volume of calls.

- For **Endpoint Type**, select **Webhook**.
- Click **Select an endpoint**. On the **Select Web Hook** page that opens up, paste the webhook url you created for the Watch-VMWrite runbook.

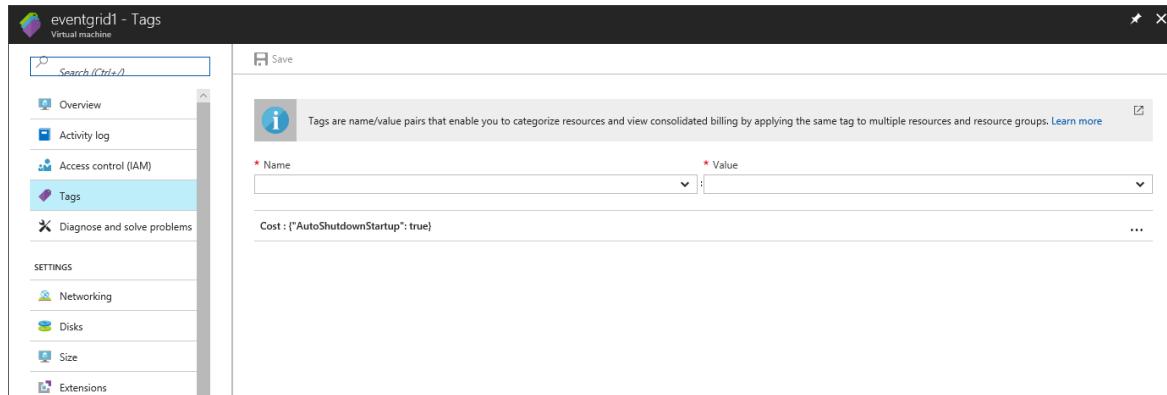
- g. Under **FILTERS**, enter the subscription and resource group where you want to look for the new VMs created. It should look like:

```
/subscriptions/<subscription-id>/resourcegroups/<resource-group-name>/providers/Microsoft.Compute/virtualMachines
```

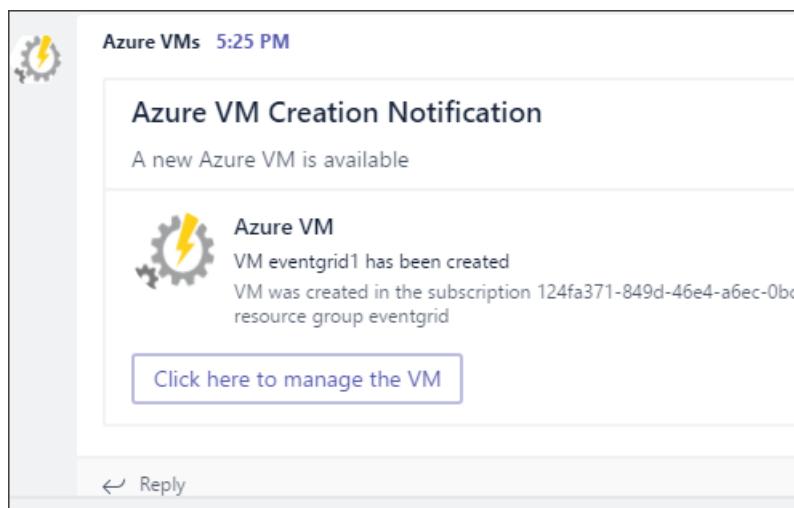
4. Select **Create** to save the Event Grid subscription.

Create a VM that triggers the runbook

1. Create a new VM in the resource group you specified in the Event Grid subscription prefix filter.
2. The Watch-VMWrite runbook should be called and a new tag added to the VM.



3. A new message is sent to the Microsoft Teams channel.



Next steps

In this tutorial, you set up integration between Event Grid and Automation. You learned how to:

- Import an Event Grid sample runbook.
- Create an optional Microsoft Teams webhook.
- Create a webhook for the runbook.
- Create an Event Grid subscription.
- Create a VM that triggers the runbook.

[Create and route custom events with Event Grid](#)

How to start and stop Azure-SSIS Integration Runtime on a schedule

9/10/2021 • 14 minutes to read • [Edit Online](#)

APPLIES TO: Azure Data Factory Azure Synapse Analytics

This article describes how to schedule the starting and stopping of Azure-SSIS Integration Runtime (IR) by using Azure Data Factory (ADF). Azure-SSIS IR is ADF compute resource dedicated for executing SQL Server Integration Services (SSIS) packages. Running Azure-SSIS IR has a cost associated with it. Therefore, you typically want to run your IR only when you need to execute SSIS packages in Azure and stop your IR when you do not need it anymore. You can use ADF User Interface (UI)/app or Azure PowerShell to [manually start or stop your IR](#).

Alternatively, you can create Web activities in ADF pipelines to start/stop your IR on schedule, e.g. starting it in the morning before executing your daily ETL workloads and stopping it in the afternoon after they are done. You can also chain an Execute SSIS Package activity between two Web activities that start and stop your IR, so your IR will start/stop on demand, just in time before/after your package execution. For more info about Execute SSIS Package activity, see [Run an SSIS package using Execute SSIS Package activity in ADF pipeline](#) article.

NOTE

This article has been updated to use the Azure Az PowerShell module. The Az PowerShell module is the recommended PowerShell module for interacting with Azure. To get started with the Az PowerShell module, see [Install Azure PowerShell](#). To learn how to migrate to the Az PowerShell module, see [Migrate Azure PowerShell from AzureRM to Az](#).

Prerequisites

If you have not provisioned your Azure-SSIS IR already, provision it by following instructions in the [tutorial](#).

Create and schedule ADF pipelines that start and or stop Azure-SSIS IR

This section shows you how to use Web activities in ADF pipelines to start/stop your Azure-SSIS IR on schedule or start & stop it on demand. We will guide you to create three pipelines:

1. The first pipeline contains a Web activity that starts your Azure-SSIS IR.
2. The second pipeline contains a Web activity that stops your Azure-SSIS IR.
3. The third pipeline contains an Execute SSIS Package activity chained between two Web activities that start/stop your Azure-SSIS IR.

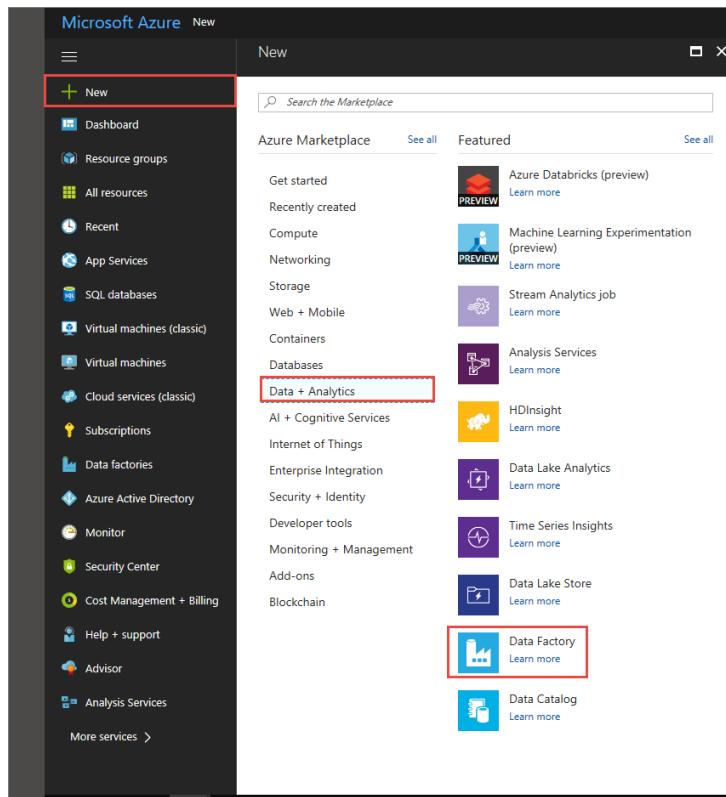
After you create and test those pipelines, you can create a schedule trigger and associate it with any pipeline. The schedule trigger defines a schedule for running the associated pipeline.

For example, you can create two triggers, the first one is scheduled to run daily at 6 AM and associated with the first pipeline, while the second one is scheduled to run daily at 6 PM and associated with the second pipeline. In this way, you have a period between 6 AM to 6 PM every day when your IR is running, ready to execute your daily ETL workloads.

If you create a third trigger that is scheduled to run daily at midnight and associated with the third pipeline, that pipeline will run at midnight every day, starting your IR just before package execution, subsequently executing your package, and immediately stopping your IR just after package execution, so your IR will not be running idly.

Create your ADF

1. Sign in to [Azure portal](#).
2. Click **New** on the left menu, click **Data + Analytics**, and click **Data Factory**.



3. In the New data factory page, enter MyAzureSsisDataFactory for Name.

The 'New data factory' dialog box contains the following fields:

- Name:** MyAzureSsisDataFactory0102
- Subscription:** (dropdown menu)
- Resource Group:**
 - Create new
 - Use existing AzureSsisRG
- Version:** V2 (Preview)
- Location:** East US
- Pin to dashboard:**
- Create** button
- Automation options** link

The name of your ADF must be globally unique. If you receive the following error, change the name of your ADF (e.g. yourusernameMyAzureSsisDataFactory) and try creating it again. See [Data Factory - Naming Rules](#) article to learn about naming rules for ADF artifacts.

Data factory name MyAzureSsisDataFactory is not available

4. Select your Azure **Subscription** under which you want to create your ADF.

5. For **Resource Group**, do one of the following steps:

- Select **Use existing**, and select an existing resource group from the drop-down list.
- Select **Create new**, and enter the name of your new resource group.

To learn about resource groups, see [Using resource groups to manage your Azure resources](#) article.

6. For **Version**, select **V2**.

7. For **Location**, select one of the locations supported for ADF creation from the drop-down list.

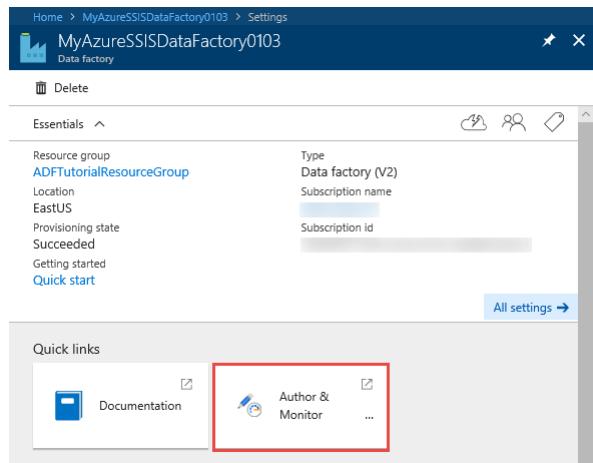
8. Select **Pin to dashboard**.

9. Click **Create**.

10. On Azure dashboard, you will see the following tile with status: **Deploying Data Factory**.



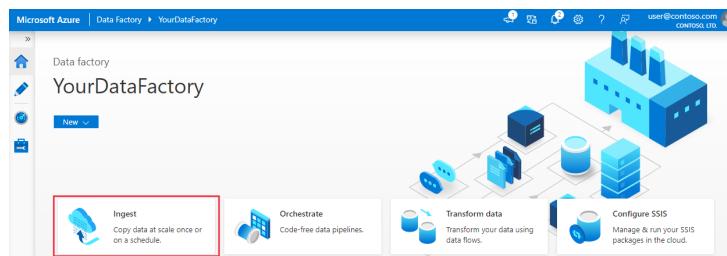
11. After the creation is complete, you can see your ADF page as shown below.



12. Click **Author & Monitor** to launch ADF UI/app in a separate tab.

Create your pipelines

1. In the home page, select **Orchestrate**.



2. In **Activities** toolbox, expand **General** menu, and drag & drop a **Web** activity onto the pipeline designer surface. In **General** tab of the activity properties window, change the activity name to **startMyIR**. Switch to **Settings** tab, and do the following actions.

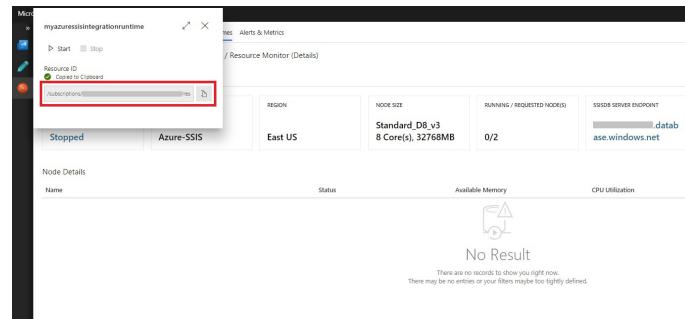
a. For **URL**, enter the following URL for REST API that starts Azure-SSIS IR, replacing

`{subscriptionId}`, `{resourceGroupName}`, `{factoryName}`, and `{integrationRuntimeName}` with the actual values for your IR:

`https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.DataFactory/factories/{factoryName}/integrationRuntimes/{integrationRuntimeName}/start?api-version=2018-06-01`

Alternatively, you can also copy & paste the resource ID of your IR from its monitoring page on ADF UI/app to replace the following part of the above URL:

`/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.DataFactory/factories/{factoryName}/integrationRuntimes/{integrationRuntimeName}/start`

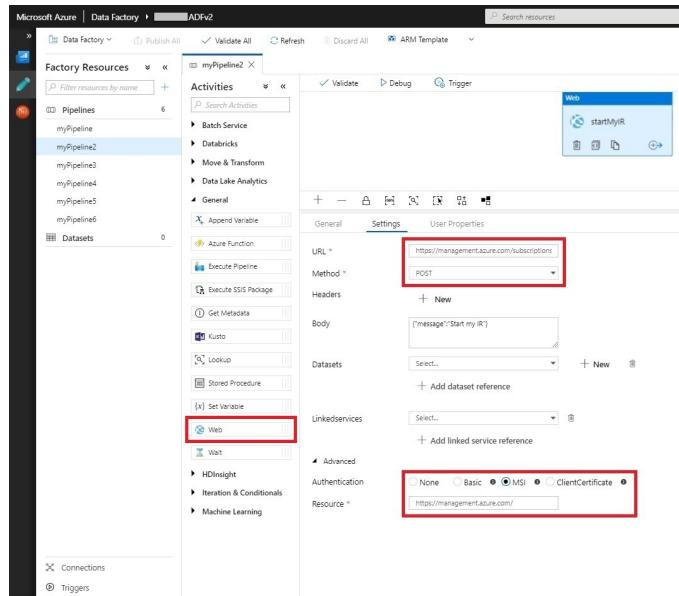


b. For **Method**, select **POST**.

c. For **Body**, enter `{"message": "Start my IR"}`.

d. For **Authentication**, select **MSI** to use the managed identity for your ADF, see [Managed identity for Data Factory](#) article for more info.

e. For **Resource**, enter `https://management.azure.com/`.



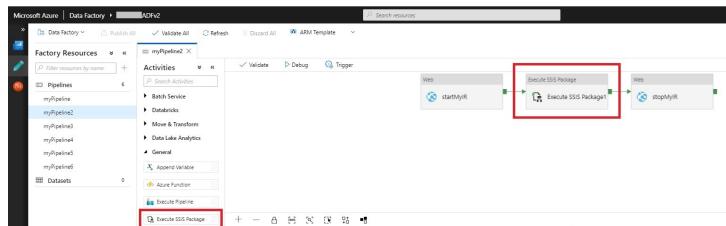
3. Clone the first pipeline to create a second one, changing the activity name to **stopMyIR** and replacing the following properties.

- a. For **URL**, enter the following URL for REST API that stops Azure-SSIS IR, replacing

{subscriptionId}, {resourceGroupName}, {factoryName}, and {integrationRuntimeName} with the actual values for your IR:
<https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.DataFactory/factories/{factoryName}/integrations/{integrationRuntimeName}?api-version=2018-06-01>

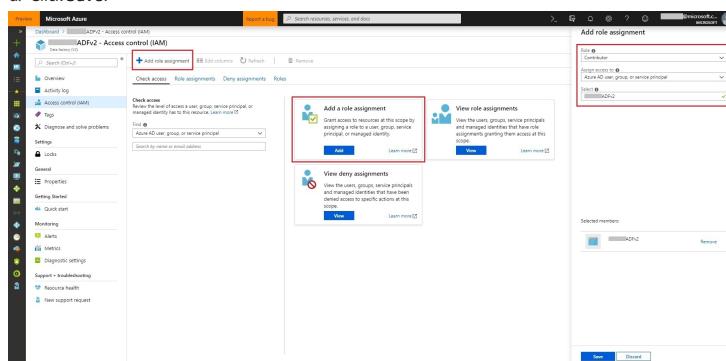
- b. For **Body**, enter `{"message": "Stop my IR"}`.

4. Create a third pipeline, drag & drop an **Execute SSIS Package** activity from Activities toolbox onto the pipeline designer surface, and configure it following the instructions in [Invoke an SSIS package using Execute SSIS Package activity in ADF](#) article. Alternatively, you can use a **Stored Procedure** activity instead and configure it following the instructions in [Invoke an SSIS package using Stored Procedure activity in ADF](#) article. Next, chain the Execute SSIS Package/Stored Procedure activity between two Web activities that start/stop your IR, similar to those Web activities in the first/second pipelines.

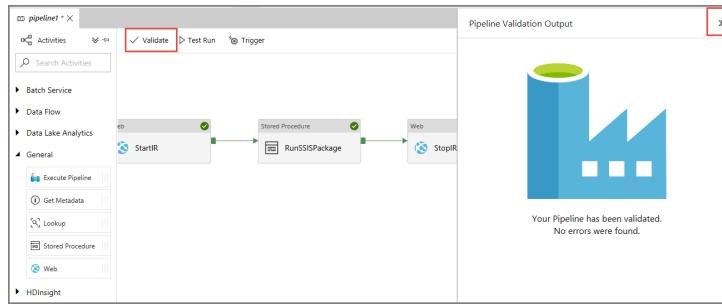


5. Assign the managed identity for your ADF a **Contributor** role to itself, so Web activities in its pipelines can call REST API to start/stop Azure-SSIS IRs provisioned in it. On your ADF page in Azure portal, click **Access control (IAM)**, click **+ Add role assignment**, and then on **Add role assignment** blade, do the following actions.

- For **Role**, select **Contributor**.
- For **Assign access to**, select **Azure AD user, group, or service principal**.
- For **Select**, search for your ADF name and select it.
- Click **Save**.



6. Validate your ADF and all pipeline settings by clicking **Validate all/Validate** on the factory/pipeline toolbar. Close **Factory/Pipeline Validation Output** by clicking >> button.



Test run your pipelines

1. Select **Test Run** on the toolbar for each pipeline and see **Output** window in the bottom pane.

Name	Type	Run Start	Duration	Status	Actions	RunID
StopIR	WebActivity	01/26/2018 5:12 PM	00:00:02	Succeeded		
RunSSISPackage	SqlServerStoredProc...	01/26/2018 5:12 PM	00:00:30	Succeeded		
StartIR	WebActivity	01/26/2018 4:41 PM	00:00:03	Succeeded		

2. To test the third pipeline, launch SQL Server Management Studio (SSMS). In **Connect to Server** window, do the following actions.

- a. For **Server name**, enter <your server name>.database.windows.net.
- b. Select **Options >>**.
- c. For **Connect to database**, select **SSISDB**.
- d. Select **Connect**.
- e. Expand **Integration Services Catalogs** -> **SSISDB** -> Your folder -> **Projects** -> Your SSIS project -> **Packages**.
- f. Right-click the specified SSIS package to run and select **Reports** -> **Standard Reports** -> **All Executions**.
- g. Verify that it ran.

ID	Status	Report	Folder Name	Project Name	Package Name	Start Time	End Time	Duration (sec)
7	Success	MySSISProjects	MySSISProjects	SSIS Test	Package.dtsx	1/26/2018 10:12:12 PM	1/26/2018 10:12:39 PM	17
8	Success	MySSISProjects	MySSISProjects	SSIS Test	Package.dtsx	1/26/2018 11:11:59 PM	1/26/2018 11:12:09 PM	11

Schedule your pipelines

Now that your pipelines work as you expected, you can create triggers to run them at specified cadences. For details about associating triggers with pipelines, see [Trigger the pipeline on a schedule](#) article.

1. On the pipeline toolbar, select **Trigger** and select **New/Edit**.

2. In **Add Triggers** pane, select **+ New**.

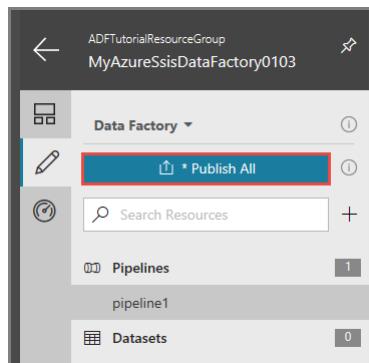
3. In **New Trigger** pane, do the following actions:

- For **Name**, enter a name for the trigger. In the following example, **Run daily** is the trigger name.
- For **Type**, select **Schedule**.
- For **Start Date (UTC)**, enter a start date and time in UTC.
- For **Recurrence**, enter a cadence for the trigger. In the following example, it is **Daily** once.
- For **End**, select **No End** or enter an end date and time after selecting **On Date**.
- Select **Activated** to activate the trigger immediately after you publish the whole ADF settings.
- Select **Next**.

The screenshot shows the 'New Trigger' dialog box. The 'Name' field contains 'Run daily'. The 'Type' section has 'Schedule' selected. The 'Start Date (UTC)' is set to '01/26/2018, 10:40 PM'. The 'Recurrence' section shows 'Daily' selected with 'Every 1 Day(s)'. Under 'Advanced recurrence options', there are fields for 'Execute at these times' (Hours and Minutes UTC), 'End' (No End selected), and 'Activated' (checkbox checked). At the bottom are 'Cancel' and 'Next' buttons.

4. In **Trigger Run Parameters** page, review any warning, and select **Finish**.

5. Publish the whole ADF settings by selecting **Publish All** in the factory toolbar.



Monitor your pipelines and triggers in Azure portal

1. To monitor trigger runs and pipeline runs, use **Monitor** tab on the left of ADF UI/app. For detailed steps, see [Monitor the pipeline](#) article.

The screenshot shows the 'Pipeline Runs' monitor page. It displays two pipeline runs for 'pipeline1': one run started at 01/26/2018, 7:30:01 PM and another at 01/26/2018, 6:30:00 PM, both of which succeeded. The 'Actions' column shows a link for each run.

Pipeline Name	Actions	Run Start	Duration	Triggered By	Status	Parameters	Error
pipeline1	View Activity Runs	01/26/2018, 7:30:01 PM	00:30:46	ScheduleTrigger	Succeeded		
pipeline1	View Activity Runs	01/26/2018, 6:30:00 PM	00:30:53	ScheduleTrigger	Succeeded		

2. To view the activity runs associated with a pipeline run, select the first link (**View Activity Runs**) in **Actions** column. For the third pipeline, you will see three activity runs, one for each chained activity in the pipeline (Web activity to start your IR, Stored Procedure activity to execute your package, and Web activity to stop your IR). To view the pipeline runs again, select **Pipelines** link at the top.

Activity Name	Activity Type	Actions	Run Start	Duration	Status	Integration Runtime
StopIR	WebActivity	[Edit] [Delete]	01/26/2018, 8:00:40 PM	00:00:05	Succeeded	
RunSSISPackage	SqlServerStoredProcedure	[Edit] [Delete]	01/26/2018, 8:00:09 PM	00:00:29	Succeeded	DefaultIntegrationRuntime (East US)
StartIR	WebActivity	[Edit] [Delete]	01/26/2018, 7:30:03 PM	00:00:02	Succeeded	

3. To view the trigger runs, select **Trigger Runs** from the drop-down list under **Pipeline Runs** at the top.

Trigger Name	Trigger Type	Trigger Time	Status	Number of pipelines	Message	Properties
Run daily	ScheduleTrigger	01/26/2018, 7:30:01 PM	Succeeded	1		[Edit]
Run daily	ScheduleTrigger	01/26/2018, 6:30:00 PM	Succeeded	1		[Edit]

Monitor your pipelines and triggers with PowerShell

Use scripts like the following examples to monitor your pipelines and triggers.

1. Get the status of a pipeline run.

```
Get-AzDataFactoryV2PipelineRun -ResourceGroupName $ResourceGroupName -DataFactoryName $DataFactoryName -PipelineRunId $myPipelineRun
```

2. Get info about a trigger.

```
Get-AzDataFactoryV2Trigger -ResourceGroupName $ResourceGroupName -DataFactoryName $DataFactoryName -Name "myTrigger"
```

3. Get the status of a trigger run.

```
Get-AzDataFactoryV2TriggerRun -ResourceGroupName $ResourceGroupName -DataFactoryName $DataFactoryName -TriggerName "myTrigger" -TriggerRunStartedAfter "2018-07-15" -TriggerRunStartedBefore "2018-07-16"
```

Create and schedule Azure Automation runbook that starts/stops Azure-SSIS IR

In this section, you will learn to create Azure Automation runbook that executes PowerShell script, starting/stopping your Azure-SSIS IR on a schedule. This is useful when you want to execute additional scripts before/after starting/stopping your IR for pre/post-processing.

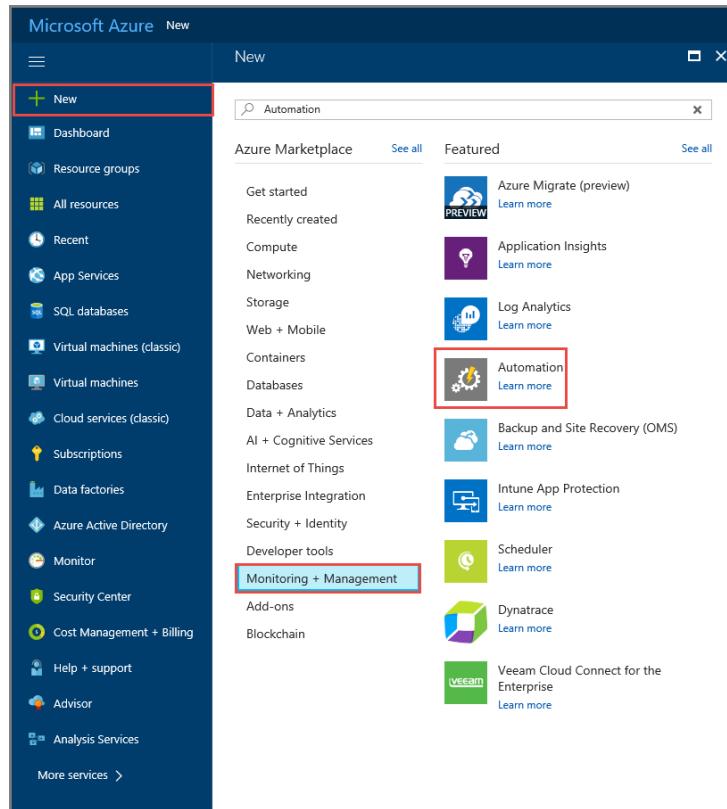
Create your Azure Automation account

If you do not have an Azure Automation account already, create one by following the instructions in this step. For detailed steps, see [Create an Azure Automation account](#) article. As part of this step, you create an **Azure Run As** account (a service principal in your Azure Active Directory) and assign it a **Contributor** role in your Azure subscription. Ensure that it is the same subscription that contains your ADF with Azure SSIS IR. Azure Automation will use this account to authenticate to Azure Resource Manager and operate on your resources.

1. Launch **Microsoft Edge** or **Google Chrome** web browser. Currently, ADF UI/app is only supported in Microsoft Edge and Google Chrome web browsers.

2. Sign in to [Azure portal](#).

3. Select **New** on the left menu, select **Monitoring + Management**, and select **Automation**.



4. In Add Automation Account pane, do the following actions.

- For **Name**, enter a name for your Azure Automation account.
- For **Subscription**, select the subscription that has your ADF with Azure-SSIS IR.
- For **Resource group**, select **Create new** to create a new resource group or **Use existing** to select an existing one.
- For **Location**, select a location for your Azure Automation account.
- Confirm **Create Azure Run As account** as **Yes**. A service principal will be created in your Azure Active Directory and assigned a **Contributor** role in your Azure subscription.
- Select **Pin to dashboard** to display it permanently in Azure dashboard.
- Select **Create**.

The dialog box is titled "Add Automation Account". It includes the following fields:

- Name**: MyAutomation
- Subscription**: (dropdown menu)
- Resource group**: Create new (radio button) selected, Use existing (radio button) unselected. Value: ADFTutorialResourceGroup
- Location**: East US 2
- Create Azure Run As account**: Yes (radio button selected, No unselected)

A note below the location field states: "The Run As account feature will create a Run As account and a Classic Run As account. Click here to learn more about Run As accounts." There is also a link to "Learn more about Automation pricing".

At the bottom, there is a checked checkbox for "Pin to dashboard" and a blue "Create" button.

5. You will see the deployment status of your Azure Automation account in Azure dashboard and notifications.



6. You will see the homepage of your Azure Automation account after it is created successfully.

Import ADF modules

- Select **Modules** in **SHARED RESOURCES** section on the left menu and verify whether you have **Az.DataFactory + Az.Profile** in the list of modules.

Name	Last Modified	Status	Version
Azure	2/7/2018, 12:08 AM	Available	5.1.1
Azure Storage	2/7/2018, 12:08 AM	Available	4.1.0
AzurermAutomation	2/7/2018, 12:08 AM	Available	4.2.0
AzurermCompute	2/7/2018, 12:08 AM	Available	4.2.0
AzurermProfile	4/7/2018, 9:21 AM	Available	1.0.3
AzurermResources	2/7/2018, 12:08 AM	Available	5.2.0
AzurermSql	2/7/2018, 12:08 AM	Available	4.2.0
AzurermStorage	2/7/2018, 12:08 AM	Available	4.2.0
Microsoft.PowerShell.Core	4/7/2018, 9:18 AM	Available	0.0
Microsoft.PowerShell.Diagnostics	4/7/2018, 9:18 AM	Available	
Microsoft.PowerShell.Management	4/7/2018, 9:18 AM	Available	
Microsoft.PowerShell.Security	4/7/2018, 9:20 AM	Available	
Microsoft.PowerShell.Utility	4/7/2018, 9:20 AM	Available	
Microsoft.WSMAN.Management	4/7/2018, 9:21 AM	Available	
Orchestrator.AssetManagement.Cards	4/7/2018, 9:21 AM	Available	1.0

- If you do not have **Az.DataFactory**, go to the PowerShell Gallery for **Az.DataFactory module**, select **Deploy to Azure Automation**, select your Azure Automation account, and then select OK. Go back to view **Modules** in **SHARED RESOURCES** section on the left menu and wait until you see **STATUS** of **Az.DataFactory** module changed to **Available**.

Name	Last Modified	Status	Version
Azure	2/7/2018, 12:08 AM	Available	5.1.1
Azure Storage	2/7/2018, 12:08 AM	Available	4.1.0
AzurermAutomation	2/7/2018, 12:08 AM	Available	4.2.0
AzurermCompute	2/7/2018, 12:08 AM	Available	4.2.0
AzurermDataFactoryV2	4/7/2018, 9:24 PM	Available	0.5.2
AzurermProfile	4/7/2018, 9:24 PM	Available	4.6.0
AzurermResources	2/7/2018, 12:08 AM	Available	5.2.0
AzurermSql	2/7/2018, 12:08 AM	Available	4.2.0
AzurermStorage	2/7/2018, 12:08 AM	Available	4.2.0
Microsoft.PowerShell.Core	4/7/2018, 9:18 AM	Available	0.0
Microsoft.PowerShell.Diagnostics	4/7/2018, 9:18 AM	Available	
Microsoft.PowerShell.Management	4/7/2018, 9:18 AM	Available	
Microsoft.PowerShell.Security	4/7/2018, 9:20 AM	Available	
Microsoft.PowerShell.Utility	4/7/2018, 9:20 AM	Available	
Microsoft.WSMAN.Management	4/7/2018, 9:21 AM	Available	
Orchestrator.AssetManagement.Cards	4/7/2018, 9:21 AM	Available	1.0

- If you do not have **Az.Profile**, go to the PowerShell Gallery for **Az.Profile module**, select **Deploy to Azure Automation**, select your Azure Automation account, and then select OK. Go back to view **Modules** in **SHARED RESOURCES** section on the left menu and wait until you see **STATUS** of the **Az.Profile** module changed to **Available**.

Create your PowerShell runbook

The following section provides steps for creating a PowerShell runbook. The script associated with your runbook either starts/stops Azure-SSIS IR based on the command you specify for **OPERATION** parameter. This section does not provide the complete details for creating a runbook. For more information, see [Create a runbook](#) article.

1. Switch to Runbooks tab and select + Add a runbook from the toolbar.

2. Select **Create a new runbook** and do the following actions:

- a. For **Name**, enter **StartStopAzureSsisRuntime**.
- b. For **Runbook type**, select **PowerShell**.
- c. Select **Create**.

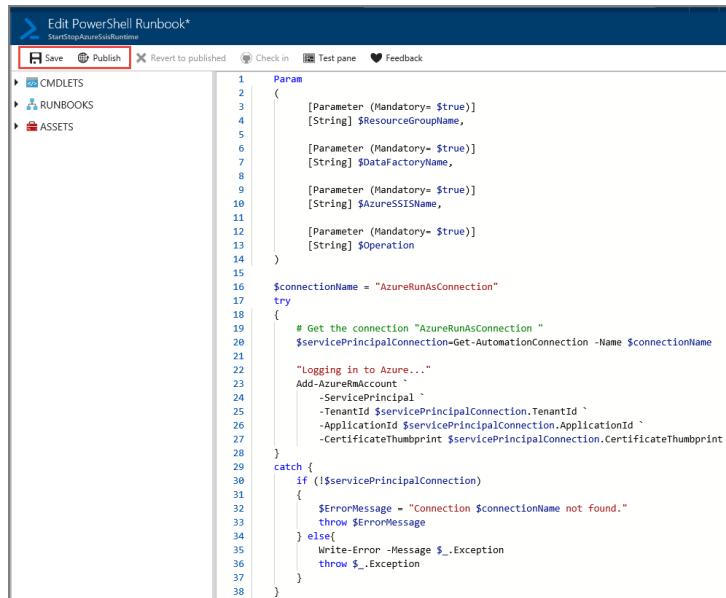
3. Copy & paste the following PowerShell script to your runbook script window. Save and then publish your runbook by using **Save** and **Publish** buttons on the toolbar.

```

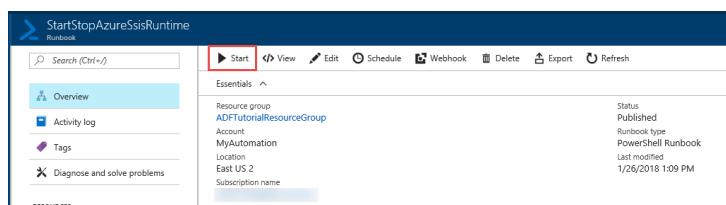
Param
(
    [Parameter (Mandatory= $true)]
    [String] $ResourceGroupName,
    [Parameter (Mandatory= $true)]
    [String] $DataFactoryName,
    [Parameter (Mandatory= $true)]
    [String] $AzureSSISName,
    [Parameter (Mandatory= $true)]
    [String] $Operation
)
$connectionName = "AzureRunAsConnection"
try
{
    # Get the connection "AzureRunAsConnection"
    $servicePrincipalConnection=Get-AutomationConnection -Name $connectionName

    "Logging in to Azure..."
    Connect-AzAccount `-
        -ServicePrincipal `-
        -TenantId $servicePrincipalConnection.TenantId `-
        -ApplicationId $servicePrincipalConnection.ApplicationId `-
        -CertificateThumbprint $servicePrincipalConnection.CertificateThumbprint
}
catch {
    if (!$servicePrincipalConnection)
    {
        $ErrorMessage = "Connection $connectionName not found."
        throw $ErrorMessage
    } else{
        Write-Error -Message $_.Exception
        throw $_.Exception
    }
}
if($Operation -eq "START" -or $operation -eq "start")
{
    "##### Starting #####"
    Start-AzDataFactoryV2IntegrationRuntime -ResourceGroupName $ResourceGroupName -DataFactoryName $DataFactoryName -Name $AzureSSISName -Force
}
elseif($Operation -eq "STOP" -or $operation -eq "stop")
{
    "##### Stopping #####"
    Stop-AzDataFactoryV2IntegrationRuntime -DataFactoryName $DataFactoryName -Name $AzureSSISName -ResourceGroupName $ResourceGroupName -Force
}
"##### Completed #####

```



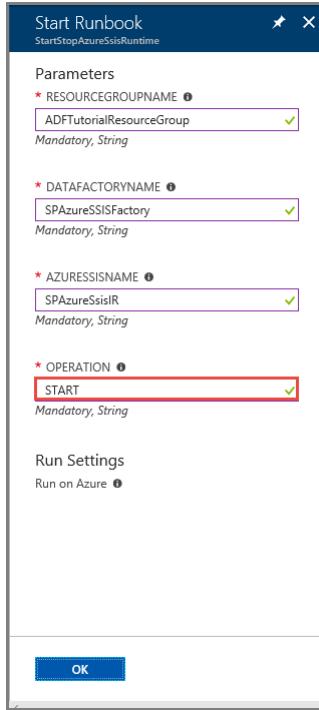
- Test your runbook by selecting **Start** button on the toolbar.



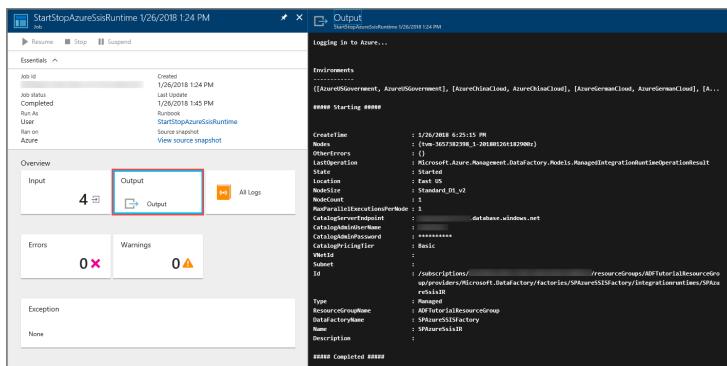
- In **Start Runbook** pane, do the following actions:

- For **RESOURCE GROUP NAME**, enter the name of resource group that has your ADF with Azure-SSIS IR.
- For **DATA FACTORY NAME**, enter the name of your ADF with Azure-SSIS IR.
- For **AZURESSISNAME**, enter the name of Azure-SSIS IR.
- For **OPERATION**, enter **START**.

e. Select OK.



6. In the job window, select **Output** tile. In the output window, wait for the message ##### Completed ##### after you see ##### Starting #####. Starting Azure-SSIS IR takes approximately 20 minutes. Close Job window and get back to Runbook window.



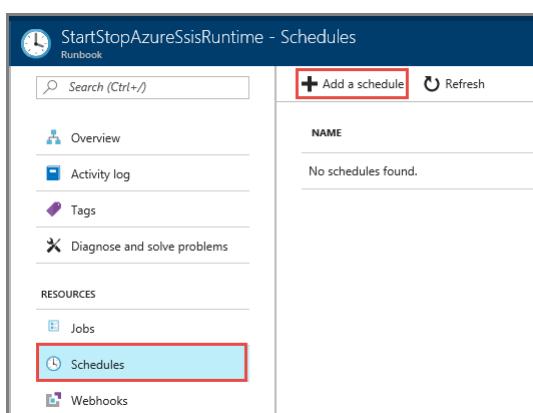
7. Repeat the previous two steps using STOP as the value for OPERATION. Start your runbook again by selecting Start button on the toolbar. Enter your resource group, ADF, and Azure-SSIS IR names. For OPERATION, enter STOP. In the output window, wait for the message ##### Completed ##### after you see ##### Stopping #####. Stopping Azure-SSIS IR does not take as long as starting it. Close Job window and get back to Runbook window.

8. You can also trigger your runbook via a webhook that can be created by selecting the **Webhooks** menu item or on a schedule that can be created by selecting the **Schedules** menu item as specified below.

Create schedules for your runbook to start/stop Azure-SSIS IR

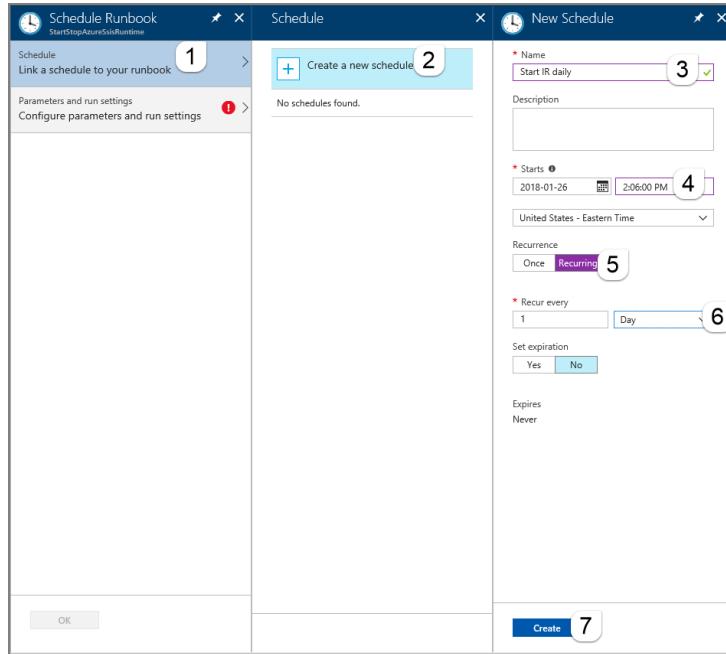
In the previous section, you have created your Azure Automation runbook that can either start or stop Azure-SSIS IR. In this section, you will create two schedules for your runbook. When configuring the first schedule, you specify START for OPERATION. Similarly, when configuring the second one, you specify STOP for OPERATION. For detailed steps to create schedules, see [Create a schedule](#) article.

1. In Runbook window, select **Schedules**, and select + Add a schedule on the toolbar.

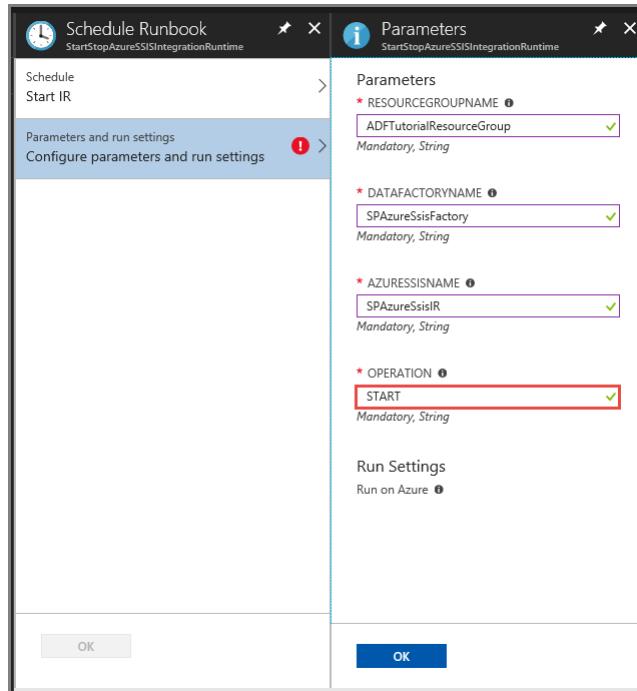


2. In Schedule Runbook pane, do the following actions:

- Select Link a schedule to your runbook.
- Select Create a new schedule.
- In New Schedule pane, enter Start IR daily for Name.
- For Starts, enter a time that is a few minutes past the current time.
- For Recurrence, select Recurring.
- For Recur every, enter 1 and select Day.
- Select Create.



3. Switch to Parameters and run settings tab. Specify your resource group, ADF, and Azure-SSIS IR names. For OPERATION, enter START and select OK. Select OK again to see the schedule on Schedules page of your runbook.



4. Repeat the previous two steps to create a schedule named Stop IR daily. Enter a time that is at least 30 minutes after the time you specified for Start IR daily schedule. For OPERATION, enter STOP and select OK. Select OK again to see the schedule on Schedules page of your runbook.

5. In Runbook window, select Jobs on the left menu. You should see the jobs created by your schedules at the specified times and their statuses. You can see the job details, such as its output, similar to what you have seen after you tested your runbook.

STATUS	CREATED	LAST UPDATED
Completed	1/26/2018 8:00 PM	1/26/2018 8:15 PM
Completed	1/26/2018 7:30 PM	1/26/2018 7:49 PM
Completed	1/26/2018 7:00 PM	1/26/2018 7:04 PM
Completed	1/26/2018 6:30 PM	1/26/2018 6:54 PM
Completed	1/26/2018 5:12 PM	1/26/2018 5:14 PM
Completed	1/26/2018 4:42 PM	1/26/2018 5:03 PM
Completed	1/26/2018 2:00 PM	1/26/2018 3:00 PM
Completed	1/25/2018 10:12 PM	1/25/2018 10:31 PM
Completed	1/25/2018 6:12 PM	1/25/2018 6:13 PM
Completed	1/25/2018 5:41 PM	1/25/2018 6:00 PM
Completed	1/25/2018 4:54 PM	1/25/2018 5:10 PM
Completed	1/25/2018 4:23 PM	1/25/2018 4:43 PM
Completed	1/25/2018 2:34 PM	1/25/2018 2:54 PM
Completed	1/25/2018 2:00 PM	1/25/2018 2:20 PM
Completed	1/25/2018 1:17 PM	1/25/2018 1:38 PM
Completed	1/25/2018 1:13 PM	1/25/2018 1:15 PM
Completed	1/25/2018 1:12 PM	1/25/2018 1:12 PM

6. After you are done testing, disable your schedules by editing them. Select **Schedules** on the left menu, select **Start IR daily/Stop IR daily**, and select **No** for **Enabled**.

Next steps

See the following blog post:

- [Modernize and extend your ETL/ELT workflows with SSIS activities in ADF pipelines](#)

See the following articles from SSIS documentation:

- [Deploy, run, and monitor an SSIS package on Azure](#)
- [Connect to SSIS catalog on Azure](#)
- [Schedule package execution on Azure](#)
- [Connect to on-premises data sources with Windows authentication](#)

Azure Policy built-in definitions for Azure Automation

9/13/2021 • 2 minutes to read • [Edit Online](#)

This page is an index of [Azure Policy built-in policy definitions](#) for Azure Automation. For additional Azure Policy built-ins for other services, see [Azure Policy built-in definitions](#).

The name of each built-in policy definition links to the policy definition in the Azure portal. Use the link in the **Version** column to view the source on the [Azure Policy GitHub repo](#).

Azure Automation

NAME (AZURE PORTAL)	DESCRIPTION	EFFECT(S)	VERSION (GITHUB)
Automation account variables should be encrypted	It is important to enable encryption of Automation account variable assets when storing sensitive data	Audit, Deny, Disabled	1.1.0
Automation accounts should disable public network access	Disabling public network access improves security by ensuring that the resource isn't exposed on the public internet. You can limit exposure of your Automation account resources by creating private endpoints instead. Learn more at: https://docs.microsoft.com/azure/automation/how-to/private-link-security .	Audit, Deny, Disabled	1.0.0

NAME	DESCRIPTION	EFFECT(S)	VERSION
Azure Automation accounts should use customer-managed keys to encrypt data at rest	<p>Use customer-managed keys to manage the encryption at rest of your Azure Automation Accounts. By default, customer data is encrypted with service-managed keys, but customer-managed keys are commonly required to meet regulatory compliance standards.</p> <p>Customer-managed keys enable the data to be encrypted with an Azure Key Vault key created and owned by you. You have full control and responsibility for the key lifecycle, including rotation and management. Learn more at [https://aka.ms/automation-cmk](..../articles/automation-automation-secure-asset-encryption.md#:~:text=Secure assets in Azure Automation include credentials, certificates, connections,,Using Microsoft-managed keys).</p>	Audit, Deny, Disabled	1.0.0
Configure Azure Automation accounts to disable public network access	<p>Disable public network access for Azure Automation account so that it isn't accessible over the public internet. This configuration helps protect them against data leakage risks. You can limit exposure of the your Automation account resources by creating private endpoints instead. Learn more at: https://aka.ms/privateendpoints.</p>	Modify, Disabled	1.0.0
Configure private endpoint connections on Azure Automation accounts	<p>Private endpoint connections allow secure communication by enabling private connectivity to Azure Automation accounts without a need for public IP addresses at the source or destination. Learn more about private endpoints in Azure Automation at https://docs.microsoft.com/azure/automation/how-to/private-link-security.</p>	DeployIfNotExists, Disabled	1.0.0

NAME	DESCRIPTION	EFFECT(S)	VERSION
Private endpoint connections on Automation Accounts should be enabled	Private endpoint connections allow secure communication by enabling private connectivity to Automation accounts without a need for public IP addresses at the source or destination. Learn more about private endpoints in Azure Automation at https://docs.microsoft.com/azure/automation/how-to/private-link-security	AuditIfNotExists, Disabled	1.0.0

Next steps

- See the built-ins on the [Azure Policy GitHub repo](#).
- Review the [Azure Policy definition structure](#).
- Review [Understanding policy effects](#).