

Azure Automation

Microsoft Azure Essentials



Michael McKeown

PUBLISHED BY
Microsoft Press
A division of Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

Copyright © 2015 Microsoft Corporation. All rights reserved.

No part of the contents of this book may be reproduced or transmitted in any form or by any means without the written permission of the publisher.

ISBN: 978-0-7356-9815-4

Microsoft Press books are available through booksellers and distributors worldwide. If you need support related to this book, email Microsoft Press Support at mppinput@microsoft.com. Please tell us what you think of this book at <http://aka.ms/tellpress>.

This book is provided “as-is” and expresses the authors’ views and opinions. The views, opinions, and information expressed in this book, including URL and other Internet website references, may change without notice.

Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in examples herein are fictitious. No association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

Microsoft and the trademarks listed at <http://www.microsoft.com> on the “Trademarks” webpage are trademarks of the Microsoft group of companies. All other marks are property of their respective owners.

Acquisitions, Developmental, and Project Editors: Alison Hirsch and Devon Musgrave

Editorial Production: nSight, Inc.

Copyeditor: Teresa Horton

Cover: Twist Creative

Table of Contents

Introduction	7
Who should read this ebook.....	7
Assumptions	7
Organization of this ebook.....	7
Conventions and features in this ebook.....	8
Acknowledgments.....	9
Errata, updates, & support	9
Free ebooks from Microsoft Press	9
Free training from Microsoft Virtual Academy.....	9
We want to hear from you.....	10
Stay in touch.....	10
Chapter 1 Introduction to Azure Automation	11
Why automation?	11
Repeatable deployment.....	12
Consistent testing configurations	12
Why Azure Automation?	12
Windows PowerShell workflow.....	13
End-to-end automation service.....	13
Off-premises redundancy backed storage.....	14
Runbook authoring and importing.....	14
Scenarios.....	14
Azure Automation pricing.....	15
Enabling Azure Automation.....	15
Creating an Azure Automation account.....	16
Chapter 2 Runbook management	19

What is a runbook?	19
Runbooks support in the Azure Management Portal.....	19
Import a runbook	20
Import a runbook from the Script Center	20
Import or export a runbook via the Azure Management Portal.....	21
Create a runbook.....	22
Create a runbook using Quick Create.....	22
Create a runbook from the Gallery	23
Author a runbook.....	26
Runbook parameters.....	29
Runbook checkpoints	29
Resume or suspend a runbook	32
Chapter 3 Assets	33
Management certificates.....	33
Azure Active Directory and automation.....	35
Azure Automation assets.....	36
Asset scope.....	37
Variable assets.....	38
Using a variable asset.....	40
Integration module assets.....	43
Importing an integration module asset	43
Integration modules versus runbooks	43
Credential assets	45
Creating a credential asset.....	46
Connection assets.....	48
Creating a connection asset.....	48

Using the Connect-Azure runbook	50
Calling the Connect-Azure runbook using certificates	51
Using Azure Active Directory without the Connect-Azure runbook	53
Schedule assets	54
Creating a schedule asset	54
Using the schedule	55
Chapter 4 Runbook deployment	57
Publishing a runbook	57
Invoking a runbook	58
Invoke from code within another runbook	58
Invoke a child runbook using inline scripts	62
Invoke a child runbook using Start-AzureAutomationRunbook	63
Use Start-ChildRunbook to start an Azure Automation job	64
Invoke a runbook manually from the Azure Management Portal	67
Invoke a runbook using a schedule asset	70
Troubleshooting a runbook	73
Use the Dashboard	73
Enable logging	74
Backing up a runbook	76
Chapter 5 Azure Script Center, library, and community	78
Windows PowerShell workflows and runbooks	78
Azure workflow execution	79
Resources	81
Chapter 6 Best practices in using Azure Automation	83
Runbooks	83
Concurrent editing of runbooks	85
Azure Automation accounts	85

Checkpoints.....	86
Assets	87
Importing integration modules	88
Credentials and connections.....	88
Schedules	88
Authoring runbooks	89
Chapter 7 Scenarios	91
Scenario: Provisioning of IaaS resources.....	92
Provisioning resources.....	92
Authentication processing.....	93
Using the New-AzureEnvironmentResourcesFromGallery runbook.....	94
Creating assets for the runbook	94
Defining parameters and variables	95
Configuring authentication	96
Processing details	97
Scenario: Maintaining and updating Azure IaaS resources	101
Summary of upgrade process	101
Using the Update-AzureVM runbook.....	102
Supporting runbooks.....	105
Install-ModuleOnAzureVM runbook.....	106
Copy-FileFromAzureStorageToAzureVM runbook.....	107
Copy-ItemToAzureVM runbook.....	108
Some final thoughts	109
About the Author	110

Foreword

I'm thrilled to be able to share these Microsoft Azure Essentials ebooks with you. The power that Microsoft Azure gives you is thrilling but not unheard of from Microsoft. Many don't realize that Microsoft has been building and managing datacenters for over 25 years. Today, the company's cloud datacenters provide the core infrastructure and foundational technologies for its 200-plus online services, including Bing, MSN, Office 365, Xbox Live, Skype, OneDrive, and, of course, Microsoft Azure. The infrastructure is comprised of many hundreds of thousands of servers, content distribution networks, edge computing nodes, and fiber optic networks. Azure is built and managed by a team of experts working 24x7x365 to support services for millions of customers' businesses and living and working all over the globe.

Today, Azure is available in 141 countries, including China, and supports 10 languages and 19 currencies, all backed by Microsoft's \$15 billion investment in global datacenter infrastructure. Azure is continuously investing in the latest infrastructure technologies, with a focus on high reliability, operational excellence, cost-effectiveness, environmental sustainability, and a trustworthy online experience for customers and partners worldwide.

Microsoft Azure brings so many services to your fingertips in a reliable, secure, and environmentally sustainable way. You can do immense things with Azure, such as create a single VM with 32TB of storage driving more than 50,000 IOPS or utilize hundreds of thousands of CPU cores to solve your most difficult computational problems.

Perhaps you need to turn workloads on and off, or perhaps your company is growing fast! Some companies have workloads with unpredictable bursting, while others know when they are about to receive an influx of traffic. You pay only for what you use, and Azure is designed to work with common cloud computing patterns.

From Windows to Linux, SQL to NoSQL, Traffic Management to Virtual Networks, Cloud Services to Web Sites and beyond, we have so much to share with you in the coming months and years.

I hope you enjoy this Microsoft Azure Essentials series from Microsoft Press. The first three ebooks cover fundamentals of Azure, Azure Automation, and Azure Machine Learning. And I hope you enjoy living and working with Microsoft Azure as much as we do.

*Scott Guthrie
Executive Vice President
Cloud and Enterprise group, Microsoft Corporation*

Introduction

This ebook introduces a fairly new feature of Microsoft Azure called Azure Automation. Using a highly scalable workflow execution environment, Azure Automation allows you to orchestrate frequent deployment and life cycle management tasks using runbooks based on Windows PowerShell Workflow functionality. These runbooks are stored in and backed up by Azure. By automating runbooks, you can greatly minimize the occurrence of errors when carrying out repeated tasks and process automation.

This ebook discusses the creation and authoring of the runbooks along with their deployment and troubleshooting. Microsoft has provided some sample runbooks after which you can pattern your runbooks, copy and modify, or use as-is to help your scripts be more effective and concise. This ebook explores uses of some of those sample runbooks.

Who should read this ebook

This ebook exists to help IT pros and Windows PowerShell developers understand the core concepts around Azure Automation. It's especially useful for IT pros looking for ways to automate their common Azure PaaS and IaaS application duties such as provisioning, deployment, lifecycle management, patching and updating, de-provisioning, maintenance, and monitoring.

Assumptions

You should be somewhat familiar with concepts behind Windows PowerShell programming as well as understand fundamental Azure provisioning and deployment. It helps if you have written and run some Windows PowerShell code, especially as it relates to the Azure PowerShell Management API. This ebook looks at some Azure Automation Windows PowerShell workflow scripts and breaks down what they are doing. If this is your first time with Windows PowerShell, it might be a real challenge for you.

This ebook assumes you have worked in some context with Azure in either the PaaS or IaaS spaces. Items such as Azure assets in the form of connections, credentials, variables, and schedules all will help you manage your Azure applications and deployments. For instance, you should know what is an Azure Virtual Machine (VM) or an Azure Cloud Service.

Organization of this ebook

This ebook includes seven chapters, each of which focuses on an aspect of Azure Automation, as follows:

Introduction to Azure Automation: Provides an overview of Azure Automation, looking at what it

involves, and the situations for which it is best suited. Shows how to enable Azure Automation and how to create an Azure Automation account, which is the highest-level root entity for all your automation objects under that account.

Runbook management: Covers how to manage runbooks, which are logical containers that organize and contain Windows PowerShell workflows. Also, learn about the concept of authentication and the role of management certificates or Azure Active Directory.

Assets: Describes the entities that runbooks can globally leverage across all runbooks in an Azure Automation account. Learn about variable, credential, connection, and schedule assets.

Runbook deployment: Discusses publishing a runbook after it has been authored and tested. Also provides some troubleshooting ideas.

Azure Script Center, library, and community: Learn more about Windows PowerShell Workflow functionality, the execution process, and how it relates to Azure Automation runbooks. Provides an overview of resources for reusable scripts that you can import into your runbooks and use wholly or in part.

Best practices: Looks at some key recommendations to optimize and maximize your use of Azure Automation.

Scenarios: Explores in-depth a few common Azure Automation scenarios that you can hopefully relate to your everyday work.

Conventions and features in this ebook

This ebook presents information using conventions designed to make the information readable and easy to follow:

- To create specific Azure resources, follow the numbered steps listing each action you must take to complete the exercise.
- There are currently two management portals for Azure: the Azure Management Portal at <http://manage.windowsazure.com> and the Azure Preview Portal at <http://portal.azure.com>. As of this writing, features related to Azure Automation are available only in the Azure Management Portal.
- Boxed elements with labels such as "Note" or "See Also" provide additional information.
- A plus sign (+) between two key names means that you must press those keys at the same time. For example, "Press Alt+Tab" means that you hold down the Alt key while you press Tab.
- A right angle bracket between two or more menu items (e.g., File Browse > Virtual Machines) means that you should select the first menu or menu item, then the next, and so on.

Acknowledgments

I'd like to thank the following people. Jeff Nuckolls, my manager at Aditi, who encouraged me to do this for personal growth. Charles Joy of Microsoft, who helped me get started with Azure Automation and took time to help me work through some tough issues. Joe Levy, who gave me some technical guidance to ensure I was both correct and current. And, my wife and faithful support, Tami, and my kids, Kyle, Brittany, Hap, Mikey, and Wiggy, who put up with me working all the time to get this done. Oh yeah, and so as not to offend any other family support, I might as well thank my Husky, SFD, and my two rabbits, Ting and Chesta.

Errata, updates, & support

We've made every effort to ensure the accuracy of this ebook. You can access updates to this ebook—in the form of a list of submitted errata and their related corrections—at:

<http://aka.ms/AzureAuto/errata>

If you discover an error that is not already listed, please submit it to us at the same page.

If you need additional support for this ebook, email Microsoft Press Support at msspinput@microsoft.com.

Please note that product support for Microsoft software and hardware is not offered through the previous addresses. For help with Microsoft software or hardware, go to <http://support.microsoft.com>.

Free ebooks from Microsoft Press

From technical overviews to in-depth information on special topics, the free ebooks from Microsoft Press cover a wide range of topics. These ebooks are available in PDF, EPUB, and Mobi for Kindle formats, ready for you to download at:

<http://aka.ms/mspressfree>

Check back often to see what is new!

Free training from Microsoft Virtual Academy

The Microsoft Azure training courses from Microsoft Virtual Academy cover key technical topics to help developers gain the knowledge they need to be a success. Learn Microsoft Azure from the true experts. Microsoft Azure training includes courses focused on learning Azure Virtual Machines and virtual

networks. In addition, gain insight into platform as a service (PaaS) implementation for IT Pros, including using PowerShell for automation and management, using Active Directory, migrating from on-premises to cloud infrastructure, and important licensing information.

<http://www.microsoftvirtualacademy.com/product-training/microsoft-azure>

We want to hear from you

At Microsoft Press, your satisfaction is our top priority, and your feedback our most valuable asset. Please tell us what you think of this ebook at:

<http://aka.ms/tellpress>

We know you're busy, so we've kept it short with just a few questions. Your answers go directly to the editors at Microsoft Press. (No personal information will be requested.) Thanks in advance for your input!

Stay in touch

Let's keep the conversation going! We're on Twitter: <http://twitter.com/MicrosoftPress>

Chapter 3

Assets

Microsoft Azure Automation assets are global resources used by runbooks to assist in common tasks and value-specific operations. Assets can be imported into modules. Types of assets include connections, credentials, schedules, variables, and integration modules. Global connections and credentials help authenticate between the Windows PowerShell workflows and Azure when the Azure Automation scripts are run against a specific Azure subscription. For instance, Microsoft published the Connect-Azure runbook, which can be used globally within an Azure Automation account to encapsulate the connection and login functionality needed to connect to Azure. Schedule assets can be linked to runbooks, allowing them to run at a specific date and time. Variable assets are used to provide runtime values for runbooks to work on specific subscriptions, as well as to control the logic within the Windows PowerShell code.

Azure Automation is incorporated into Azure Active Directory (Azure AD), which allows simpler management of identity and access for users and groups to the Azure Automation accounts and runbooks. Authentication can now be done with an account within Azure AD instead of having to manage and use management certificates. Using Azure AD greatly simplifies the process of authentication over using management certificates. The account in Azure AD can also be reused and leveraged in other Azure services that support the use of Azure AD.

Management certificates

To run Windows PowerShell Workflow scripts from Azure Automation, you first have to authenticate during the connection using Windows PowerShell credentials or a certificate. You must connect in an authenticated manner to Azure to be able to run any commands against resources within a subscription. Authentication must be set up between Azure Automation and the Azure resources in an Azure subscription that you intend to manipulate via script. You can upload a management certificate to handle this authentication within an Azure subscription.

Azure uses X.509 v3 certificates for authentication in many places. These certificates can be self-signed (usually done for development or testing) or signed by a trusted signature authority (usually done for production). Typically, you upload a .cer file as a management certificate. Certificates used by Azure can contain a private or a public key. A .cer management certificate file does not contain the private key embedded within it, as does a .pfx service certificate (a .pfx file is used to secure client calls to cloud services). Certificates have a thumbprint that provides a means to identify them in an unambiguous way to Azure. For a .cer file, the client connecting the service needs to be trusted and have the private key.

You can share certificates across Azure subscriptions with different subscription owners. This helps you to limit the actual number of certificates you have to create in an enterprise subscription. The limit is 100 certificates per subscription.

A management certificate is not an automation asset per se, although it is global to the subscription in its scope. You upload the management certificate just like any other management certificate in Azure, such as certificates used for Azure Recovery Services, via the Management Certificates tab under Settings.

However, for Azure Automation, the management certificate is also uploaded as an Azure Automation Credential asset if you choose to authenticate using the Certificate Credential asset. This is a key point: To work correctly for Azure Automation, a management certificate has to exist both in the Settings for the subscription and be created as a Certificate automation asset. Why the certificate needs to exist concurrently in two different forms at once at first might seem very confusing.

The Certificate Creation Tool (Makecert.exe) that ships with the Windows SDK provides information about how to create a self-signed management certificate. You can also create one using Internet Information Services (IIS). Alternatively, you can obtain a signed certificate from a verified certificate authority. However, authenticating with a certificate is no longer recommended for Azure Automation.

See Also For more information about *Makecert.exe*, see [Makecert.exe \(Certificate Creation Tool\)](#). For more information about using IIS to create a self-signed management certificate, see [Create a Self-Signed Server Certificate in IIS 7](#).

After you have the management certificate file (.cer) that contains the public key, you must upload it to Azure. Sign into the Azure Management Portal, click Settings, and then click Management Certificates. Click Upload, and then in the Upload A Management Certificate dialog box, browse to the location of your .cer file and select it. As shown in Figure 3-1, select the subscription to which you want to apply the certificate file, and then click the check mark to upload it to the Azure Management Portal.

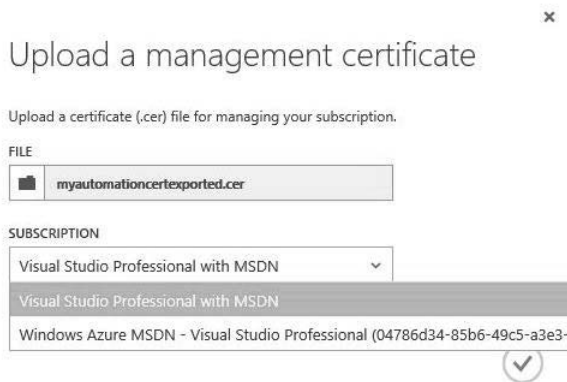
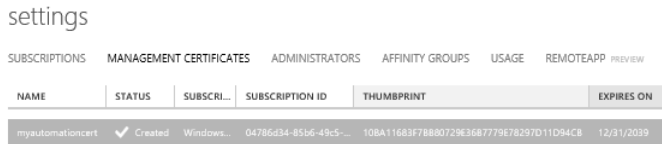


FIGURE 3-1 Dialog box to upload a management certificate to the Azure Management Portal.

After the upload completes, the certificate is displayed in the list of management certificates, as shown in Figure 3-2. The thumbprint is the public key component of the certificate. It's used with the private key component and verified against any of the loaded certificates for the subscription when Azure Automation is making requests to Azure.



The screenshot shows the 'settings' page in the Azure Management Portal. At the top, there are navigation tabs: SUBSCRIPTIONS, MANAGEMENT CERTIFICATES, ADMINISTRATORS, AFFINITY GROUPS, USAGE, and REMOTEAPP PREVIEW. Below the tabs is a table with the following columns: NAME, STATUS, SUBSCRI..., SUBSCRIPTION ID, THUMBPRINT, and EXPIRES ON. A single row is visible in the table with the following data: myautomationcert, Created, Windows..., 04785d34-85b6-49c5..., 108A11683F7880729E368779E78297D11D94CB, and 12/31/2039.

NAME	STATUS	SUBSCRI...	SUBSCRIPTION ID	THUMBPRINT	EXPIRES ON
myautomationcert	Created	Windows...	04785d34-85b6-49c5...	108A11683F7880729E368779E78297D11D94CB	12/31/2039

FIGURE 3-2 Settings section of the Azure Management Portal showing uploaded management certificates.

After you have loaded a management certificate into Azure, you're ready to create a certificate.

Azure Active Directory and automation

Authenticating using management certificates is the original and primary way to secure your calls from your Azure Automation scripts into the Azure environment, but there are a lot of steps to create and upload the certificates to Azure. Managing them can also require a lot of organizational effort.

There is now a new and recommended option that provides a more integrated and simpler authentication mechanism for Azure Automation runbooks. Using Azure AD, you can use credential-based authentication for your Azure Automation runbooks. Azure Automation allows a robust and rich, integrated, identity-based authentication mechanism, supporting key industry-wide identity access mechanisms such as single sign-on (SSO) and Multifactor Authentication (MFA). Azure Automation easily integrates and synchronizes with your on-premises enterprise Active Directory installation. Azure Automation also uses role-based access control (RBAC) mechanisms available in the Azure Preview Portal. Additionally, you can leverage RBAC in your Azure Automation runbook authentication strategy. This permits you to simplify and improve control regarding who in your organization is allowed to perform specific operations or access specific resources.

Azure Automation is becoming increasingly integrated into the various Azure services as an all-inclusive identity solution. With Azure Automation, your organizational groups and user accounts are used to simplify secure access to different parts of Azure. When you log into your Azure subscription or use the Azure REST Management application programming interface (API), you authenticate using Azure Automation. Azure Automation, along with services such as Microsoft Office 365, Microsoft Azure SQL Database, Microsoft Azure Mobile Services, and Microsoft Azure Cloud Services, trust Azure Automation with identity access management.

To enable Azure Automation for a new user, do the following:

1. Create the user in Azure AD. For more information about creating a user in Azure AD, see [Create or edit users](#).

2. Add the user as co-administrator to your Azure subscription. Log in to the Azure Management Portal at *manage.windowsazure.com*, click Settings, click Administrators, and then click Add.
3. Log in to the Azure Management Portal as the Azure Automation user you created in step 1 and change the password when prompted.

(This procedure isn't necessary if you want to use an existing Azure user account.) After the user is created, you will want to create an Azure Automation credential asset with the login credentials of that user. As a best practice, it often makes sense to create a user account just to use for running your Azure Automation scripts.

You can access the Azure Automation credential asset from within your Azure Automation runbook. The runbook code gets the credentials from Azure Automation, using the Azure Automation credential asset, and then uses the credentials to authenticate when it connects to Azure.

In the following example, Kim Akers is the credential asset used to authenticate with Azure AD. The Windows PowerShell workflow code makes a call to the `Get-AutomationPSCredential` cmdlet to authenticate the script:

```
Workflow Get-AzureVMNamesSample
{
    # Grab the credential to use to authenticate to Azure.
    # TODO: Fill in the -Name parameter with the name of the Automation PSCredential asset
    # that has access to your Azure subscription
    $Cred - Get-AutomationPSCredential -Name "KimAkers.onmicrosoft.com"

    # Connect to Azure
    Add-AzureAccount -Credential $Cred

    InlineScript {

        # Select the Azure subscription you want to work against
        # TODO: Fill in the -SubscriptionName parameter with the name of your Azure subscription
        Select-AzureSubscription -SubscriptionName "Windows Azure MSDN - Visual Studio Ultimate"

        # Get all Azure VMs in the subscription, and output each VM's name
        Get AzureVM | select InstanceName
    }
}
```

Although using management certificates to authenticate Azure Automation runbooks is still supported, as a best practice, use Azure AD for all your Azure Automation authentication mechanisms whenever possible.

Azure Automation assets

Assets are to Azure Automation as running water is to a modern home. Sure, you can exist without

piped running water by going to the stream or lake near your home (if you have one), manually filling buckets of water, and lugging them home over and over again. But the spillage and time lost in this process makes it not nearly as effective as turning on the faucet to access clear and safe water out of the tap. After you have water, you use it for household tasks such as washing dishes after dinner, running the clothes washer after football practice, bathing the children in the tub, and making lemonade drink mix for snack time.

Assets serve a very similar purpose in Azure Automation as the modern day public water system. Assets are reusable shared global resources that support global and common connections, credentials, variables, and schedules. These can be shared across runbooks in the same Azure Automation account, or between multiple jobs from the same runbook. They can also manage a value from the Azure Management Portal or the Windows PowerShell command line that can be shared across runbooks. Assets promote centralized management of constant values. In the Automation area of the Azure Management Portal, assets are also referred to as settings. You can create variables that can be input by the administrator of the scripts at runtime or set via code. Assets allow a simple standard mechanism for sharing of global entities between jobs, such as variables, schedules, credentials, and connections. By using assets to encapsulate connections and credentials, the login security information is much safer than being hard-coded in workflow code. Schedule assets provide a global scheduling capability.

A good example of using assets is the Connection asset. It allows you to group the connection data necessary to connect an external system into a single object so that it can be easily accessed by runbooks. It provides a template describing how a connection for a certain system should look. This allows users to use this template when defining the connection to this system. Any changes to the connection data can be made in a single place without having to replicate the change in multiple locations (variable assets, runbooks, and so on).

Assets are useful for keeping your configuration values consistent across all runbooks. Using assets simplifies runbook maintenance by storing and maintaining configuration values in a central location. You will most likely want to use assets across multiple runbooks, so allowing updates in one place ensures the changes are reflected everywhere they are used.

Asset scope

The scope of assets is global within an Azure Automation account and shared across all runbooks in that account. For an example, see Figure 3-3. If we have a variable in Asset 1, a credential in Asset 2, and a schedule in Asset 3, with runbooks A and B in the same Azure Automation account AA, either runbook can use Assets 1, 2, and 3. When accessed in code, both runbooks get the same value for all the assets in their respective scripts. If the value is changed in runbook A, the change will be reflected in runbook B the next time it is accessed. However, runbooks in another Automation account (say BB) but in the same subscription will not have scope for any of the assets in Automation account AA.

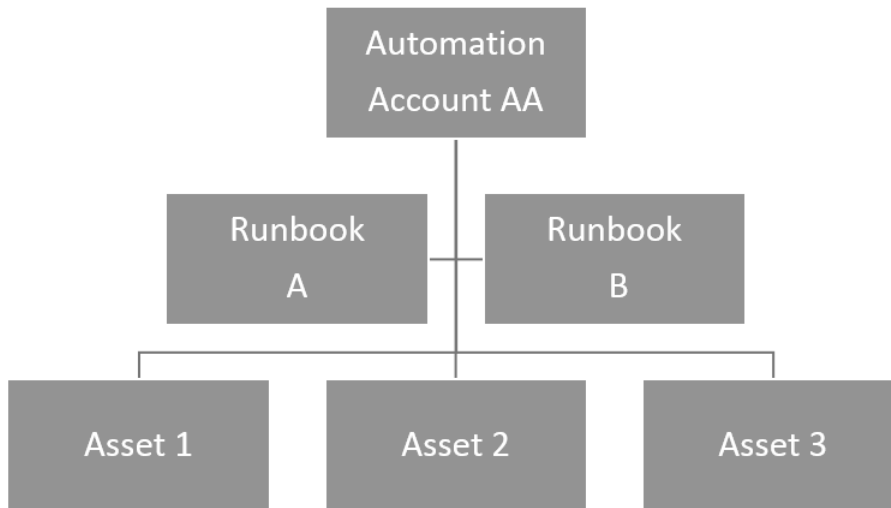


FIGURE 3-3 Runbook scope of assets within an Azure Automation account.

You can view all the assets you have for a particular Azure Automation account. Log in to the Azure Management Portal, click Automation, select the Azure Automation account, and then click Assets. Figure 3-4 shows each of the different types of assets: certificate, connection, credential, module, schedule, and variable.

[DASHBOARD](#)
[RUNBOOKS](#)
[ASSETS](#)
[SCALE](#)

NAME	TYPE	LAST UPDATE
mysamplecred	Certificate	10/18/2014 10:34:42 PM
psightcertcred	Certificate	8/19/2014 10:50:17 AM
psightconnection	Connection	8/19/2014 11:00:02 AM
mysampleconnection	Connection	10/18/2014 10:56:26 PM
psightpshellcred	Credential	8/19/2014 9:49:43 AM
Azure	Module	10/22/2014 4:21:15 PM
mysamplesched	Schedule	10/18/2014 11:36:30 PM
mystring	Variable	8/12/2014 9:07:43 PM
mysamplestring	Variable	10/18/2014 1:24:00 PM

FIGURE 3-4 Automation assets for a specific Azure Automation account.

Variable assets

Within Azure Automation, variable assets play an important part in the Windows PowerShell Workflow scripts in the runbooks. A variable is nothing more than a name that represents a value. We can use

variables to reflect changing or current values rather than entering hard-coded data directly into the script code. When the script is run, the variables are replaced with real values. This makes variables quite flexible in that they can hold and reflect data that could be different each time the runbook is run.

A variable is an asset you define that has global scope (as do all types of assets) across all runbooks in that Azure Automation account. There are four types of variables—string, integer, Boolean, datetime—and a special class named Not Defined that is basically a null value. For all types but Not Defined, you can define an initial value.

To create a variable asset, do the following:

1. Log in to the Azure Management Portal, click Automation, select the Azure Automation account, click Assets, and then click Add Settings.
2. In the Add Settings dialog box, options are available to add a connection, credential, variable, or schedule. Click Add Variable to open the Define Variable dialog box.

ADD VARIABLE

Define Variable

VARIABLE TYPE

String

NAME

mysamplestring

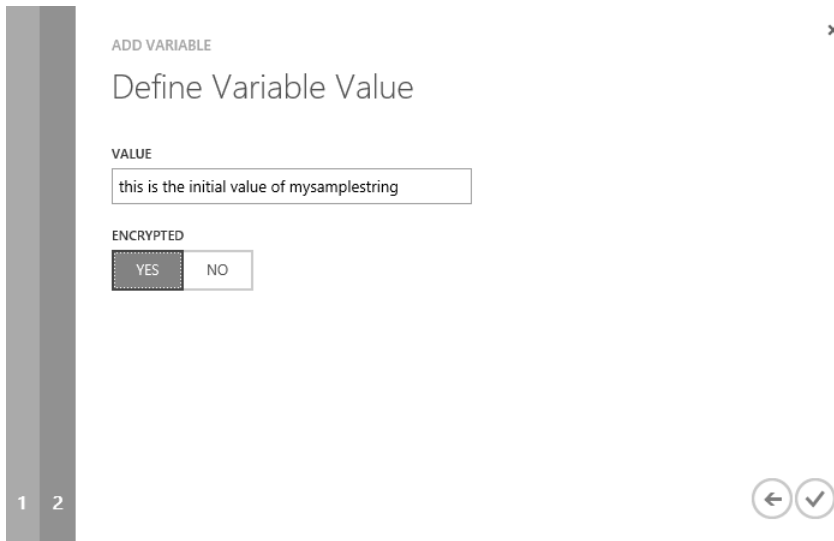
DESCRIPTION

A sample string

1

3

3. Click Variable Type and, then select String. In the Name text box, enter a name for the variable. Enter a description (this is optional). Click the right arrow to go to the Define Variable Value dialog box.



4. In the Value text box, enter the initial value for the string. This is optional, as you can leave the variable uninitialized at the start if you choose and later assign it a value at runtime or through script code. After you enter the value, you can choose to encrypt the variable. Select No if you want to see the value of the variable in the Azure Management Portal. Select Yes if you do not want to see the value of the variable in the portal. When a value is encrypted, it's displayed with circle symbols in the portal instead of its actual characters. However, it does not encrypt the data in storage. Only the appearance in the UI is encrypted. Click the check mark to create the variable.

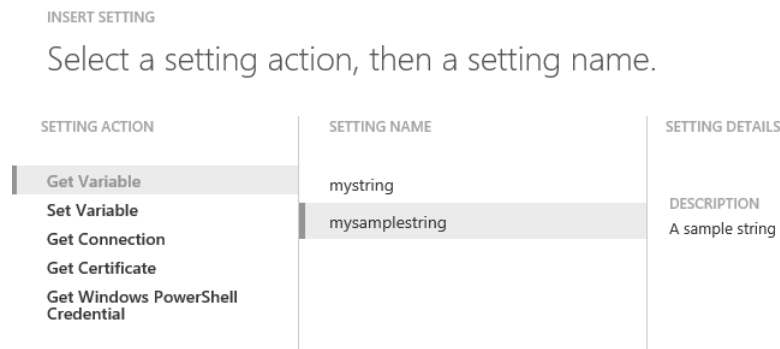
Using a variable asset

To use variable assets in a runbook, you must assign (Set) them a value in code or access (Get) their current value to do something with that value. For example, use the Set-AutomationVariable activity to set the value of a variable asset. Correspondingly use the Get-AutomationVariable activity to get the value of a variable asset. Both of these activities are part of the core Azure module, as are most of the Azure PowerShell activities used in this book.

Note For example purposes, this book uses a runbook named Demobook.

1. To set a variable value, log in to the Azure Management Portal, click Automation, click the Automation Account, click Runbooks, and then click the runbook of interest. Click the Author tab to edit that runbook.
2. Move the cursor to the location where you want to make the insertion in the Demobook runbook, and then click Insert > Setting.
3. In the Insert Setting dialog box, select a setting action (Get or Set), and then select a setting

name. This example is about setting a variable value, so select Get Variable. You can use an existing variable and set a value for it, or if the variable does not exist, create one. Choose a setting name for an existing variable. Setting Details shows the current value for that variable.



- Click the check mark to insert the Get-AutomationVariable activity into the runbook code at the location of the cursor. By default, if you don't move the cursor, the activity is inserted in the first column in the first row. If you insert a setting at that location, it will remove the name of your runbook.

demobook

DASHBOARD JOBS AUTHOR SCHEDULE CONFIGURE

PUBLISHED DRAFT

```

1 workflow Demobook
2 {
3   Get-AutomationVariable -Name 'mysamplestring'
4 }
```

To insert the Set-AutomationVariable activity, use the same process except choose that activity from the Setting Action column in the Insert Setting dialog box.

You can use the Get-AutomationVariable and Set-AutomationVariable activities together to understand the concept of global scope for assets. The following process uses the mysamplestring variable asset and the Demobook runbook shown previously. In addition, a second runbook example named Demobook2 shows the global scope across runbooks of a variable asset.

- Create a temporary variable \$testValue, and then assign it the value of mysamplestring. Make a call to write-output to display the original value of \$testvalue. Click Test to run and show this output.

demobook

 DASHBOARD [JOBS](#) [AUTHOR](#) [SCHEDULE](#) [CONFIGURE](#)


PUBLISHED [DRAFT](#)

```
1 workflow Demobook
2 {
3   #print out original value of global asset mysamplestring
4   $testValue = Get-AutomationVariable -Name 'mysamplestring'
5   write-output $testValue
6 }
```

2. Create a new runbook called Demobook2.
3. Assign a new value to mysamplestring of “new value for mysamplestring”. Click Insert >Setting. In the Insert Setting dialog box, under Setting Action, select Set Variable. In Setting Name select the setting name, and then click the check mark. This action results in the following being written to the Demobook runbook at the current cursor location:

Set-AutomationVariable -Name 'mysamplestring' -Value <System.Object>

4. Replace the <System.Object> with \$newmysamplestring. This sets the value of mysamplestring to the value contained in \$newmysamplestring. Call Get-AutomationVariable to obtain the value of mysamplestring into the \$testvalue variable, which has just changed. Call write-output to display the value of \$testvalue. Click Test to run the code and see the output of both the original global value of mysamplestring of “mysamplestring” and the updated global value of “new value for mysamplestring”.

 DASHBOARD [JOBS](#) [AUTHOR](#) [SCHEDULE](#) [CONFIGURE](#)

PUBLISHED [DRAFT](#)

```
1 workflow Demobook2
2 {
3   #assign new value to global asset mysamplestring
4   $newmysamplestring = "new value for mysamplestring"
5   Set-AutomationVariable -Name 'mysamplestring' -Value $newmysamplestring
6   $testValue = Get-AutomationVariable -Name 'mysamplestring'
7   write-output $testValue
8 }
```

 OUTPUT PANE

STATUS: COMPLETED

```
new value for mysamplestring
```

This example demonstrates that all runbooks in an Azure Automation account share the same global value for mysamplestring. If one runbook changes its value, the change is reflected across all runbooks in that automation account. Also note that if you have a variable asset in another of your automation accounts by the same name—mysamplestring, in this case—it will be a completely

different value and in a totally different storage location than the `mysamplestring` variable in your other runbook.

This principle applies not just to variable assets, but to other assets you can insert into code, including connections, certificates, and Windows PowerShell credentials. Schedule assets are a bit different from the other types of assets in that you don't call them in scripts. However, their capability is still global to all runbooks in an Azure Automation account.

Integration module assets

Integration modules are published Windows PowerShell libraries that can be imported into Azure Automation as a module asset and used when authoring runbooks. They can be up to 30 MB in size and must be in zipped format. By default, when you create an Automation Account, the Azure PowerShell module is imported. This module asset supplies all the Azure PowerShell cmdlets (also referred to as activities) that you can use in your runbooks. You can see the Azure module by itself initially when an Azure Automation account is created. Additionally, you can import additional Windows PowerShell Workflow modules as assets.

Importing an integration module asset

To import a module asset, do the following:

1. Download the module asset as a .zip file and then save it locally.
2. In the Azure Management Portal, click Automation, select the Azure Automation account, and then click Assets.
3. Click Import Module to browse and select the module to be imported, and then click the check mark to begin the import process. The display shows it is unzipping the activities in that module. After the module has completed the import process, it is displayed at the Azure Automation Account level under Assets as a Module asset type.

The most common issue encountered during importing a module is that the zipped module package must contain a single folder within the .zip file that has the same name as the .zip file. Within this folder, there needs to be at least one file with the same name as the folder, and using the extension .psd1, .psm1, or .dll. Also, the Integration Module package is a compressed file with the same name as the module and a .zip extension. It contains a single folder also with the name of the module. The Windows PowerShell module and any supporting files, including a manifest file (.psd1) if the module has one, must be contained in this folder.

Integration modules versus runbooks

A common misunderstanding in Azure Automation is the concept of a module versus a runbook, as well as the terms *import* and *insert*.

An Azure Automation *runbook* is an execution unit that contains Windows PowerShell Workflow

Parameters for 'Add-AzureDisk'

PARAMETERS		
NAME	TYPE	REQUIRED
DiskName	System.String	Yes
Label	System.String	No
MediaLocation	System.String	Yes
OS	System.String	No

FIGURE 3-6 The parameters for the Add-AzureDisk activity.

When you click the check mark to close this dialog box, a template for Add-AzureDisk is added to the cursor location for the runbook. In the following code example, note the line continuation character ``` at the end of each line as it is inserted. Azure uses this method to insert each activity into a script. If you prefer, you can remove these characters and put the command all on one line.

```
Add-AzureDisk `
-DiskName <System.String> `
  -MediaLocation <System.String> `
  [-Label <System.String>] `
  [-OS <System.String>]
```

Due to the lack of brackets `[]` around them, the first two parameters, `DiskName` and `MediaLocation`, are required when using this activity. The other two parameters in `[]` square brackets, `Label` and `OS`, are optional. You would replace the `<System.String>` entities with actual string values or temporary variables. For instance, the call within your runbook at runtime might look something like the following example:

```
Add-AzureDisk -DiskName "MyDiskName" -MediaLocation
"http://mystorageaccount.blob.core.azure.com/vhds/winserver-system.vhd" -Label "BootDisk" -OS
"Windows"
```

Credential assets

The credential asset is used to gain secure access to external systems, networks, databases, services, and so on. You can view it as a "Run As" security principal that gives an identity to the call into that external system. This asset is used most commonly in IaaS deployment situations such as authenticating when accessing a SharePoint or a SQL Server IaaS VM. Credential properties are stored in Azure Automation assets and are referenced inside of script workflows using either the `Get-AutomationCertificate` or the `Get-AutomationPSCredential` activity.

When using credential assets, you can authenticate using either a certificate credential or a

Windows PowerShell credential. Certificate credentials are based on a management certificate. It's a best practice to use Azure AD for the certificate. The connection asset uses the management certificate to authenticate to that subscription. A Windows PowerShell credential uses the script when it connects to the VM and needs to provide a username and a password. Typically this identity is used to log into an Azure VM.

Creating a credential asset

To create a credential asset, do the following:

1. In the Azure Management Portal, click Automation, select the Azure Automation account, and then click Assets. Click Add Setting.
2. In the Add Setting dialog box, click Add Credential.
3. In the Add Credential dialog box, select the Credential Type of the setting you want to add. In the following screenshot, the Certificate credential type is selected. The other credential type option is Windows PowerShell Credential, where the user will need to provide the userid and password credentials. In addition, provide a name and brief description in the Name and Description text boxes. After you have provided the information, click the right arrow.

ADD CREDENTIAL

Define Credential

CREDENTIAL TYPE

Certificate

NAME

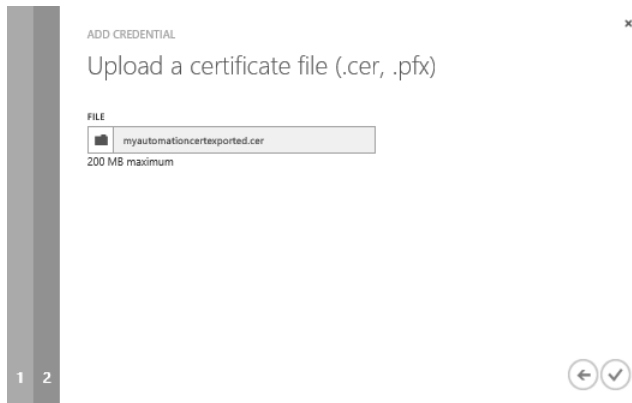
mysamplecred

DESCRIPTION

A sample credential for demo purposes

1

4. On the Upload A Certificate File page, browse for a certificate file, which can be a .cer or .pfx file.



After you load the certificate and create the credential, you can go to the Assets tab, find your new credential, open it, and see the certificate details, as shown in Figure 3-7. Note the value of the thumbprint.

mysamplecred

certificate details

NAME	mysamplecred
LAST UPDATE	10/18/2014 10:34:42 PM
CREDENTIAL TYPE	Certificate
THUMBPRINT	10BA11683F7B880729E3687779E78297D11D94CB
DESCRIPTION	A sample credential for demo purposes

FIGURE 3-7 Details of a certificate that has been installed in Azure to support the credential asset.

If you go into the management certificate section of the Azure Management Portal and find the certificate you just loaded up for this credential, you will see that same thumbprint value. When Azure Automation tries to authenticate, it will use this credential to access the thumbprint and pass it to Azure. Azure will attempt to match the thumbprint of the credential to that of the corresponding certificate to authenticate the call.

ADD CONNECTION

Configure connection

CONNECTION TYPE
 Azure

NAME
 mysampleconnection

DESCRIPTION
 Connection for demo work

1

5. Click the right arrow to configure the connection properties. Here you need to add the name of the Azure Automation certificate and the subscription ID that you previously copied into Notepad. Click the check mark to complete the creation.

ADD CONNECTION

Configure connection properties

AUTOMATIONCERTIFICATENAME
 myautomationcert

SUBSCRIPTIONID
 04786d34-85b6-49c5-a3e3-564d625e1aa1

1 2

When you're done with this process, open the connection you just created to view the connection details. On the Connection Details page (see Figure 3-8), you find the name of the connection, the day and time it was last updated, its type, a description, the subscription ID, and the Automation certificate name.

mysampleconnection

connection details

NAME mysampleconnection

LAST UPDATE 10/18/2014 10:56:26 PM

TYPE Azure

DESCRIPTION Connection for demo work

SUBSCRIPTIONID 04786d34-85b6-49c5-a3e3-...

AUTOMATIONCERTIFICATENAME myautomationcert

FIGURE 3-8 Properties of an established connection.

Using the Connect-Azure runbook

Microsoft has created a collection of sample runbooks and published them for free public download and use in the MSDN Library at [Sample runbooks for Azure Automation](#). The Connect-Azure runbook is one of the most commonly used runbooks. This runbook is used to connect to an Azure subscription. You will probably want to import it into most of your runbooks that connect to Azure to do operations. A script that is running while hosted in an Azure-managed VM process must connect to your Azure subscription to access the resources you are trying to manipulate with the script.

You can import the Connect-Azure runbook into your Azure Automation account, and then publish it so other runbooks can call it. After it's imported, the Connect-Azure runbook can be a key part of your connectivity, leveraging assets for Azure Automation. This is because it uses the Azure connection and credential assets, which you need for any Azure Automation connection. It inserts the Azure Management certificate from the local machine store to set up a connection to the Azure subscription.

Before you use this runbook, you must make sure that you have an Automation certificate asset in Azure that holds the management certificate. You must also have a connection asset containing the name of the certificate and the subscription ID.

Before we talk in more detail about this runbook, recall that we mentioned earlier the ability to authenticate now with Azure AD and not have to use management certificates. At the end of this section I discuss a bit about how to do that. However, the concepts shown in the Connect-Azure runbook are still applicable and are great examples of how to use the credential and connection assets

together to authenticate the runbook to Azure. It also is a very good example of how to call an imported runbook from another runbook.

Calling the Connect-Azure runbook using certificates

The Connect-Azure runbook takes a connection name as a parameter. This parameter can be passed on the command line if you were invoking the connect-azure script from the command line. However, as it is an imported module, you will most likely call it from another runbook, passing in the AzureConnectionName parameter, which is an Azure Automation connection asset. The connection asset is a common connection object that you can define as an Automation asset in the Azure Management Portal to be used globally by many runbooks. When you create a connection asset, you specify the subscription ID. In addition, a certificate asset contains the management certificate that is correlated to that connection asset.

In the following code example, the Param block shows the mandatory (since `=$true`) string parameter that accepts the name of the connection. Several Get PowerShell commands take strings and output automation objects. In this case, the Connect-Azure runbook gets back an Automation connection object. The call to Get-AutomationConnection is wrapped in an exception block that will throw an exception and end the processing of the script if it can't find the named connection. After it gets the connection object, the script accesses the AutomationCertificateName property, again throwing an exception if it is unable to access that value. If the script can access the property, an AutomationCertificate object is returned. After the script has both the Automation connection object and the certificate object, the script then calls the Set-AzureSubscription cmdlet, passing in the connection name, the subscription ID (acquired from the connection object), and the certificate object.

```
workflow Connect-Azure
{
    Param
    (
        [Parameter(Mandatory=$true)]
        [String]
        $AzureConnectionName
    )

    # Get the Azure connection asset that is stored in the Automation service based on the name that
    was passed into the runbook
    $AzureConn = Get-AutomationConnection -Name $AzureConnectionName
    if ($AzureConn -eq $null)
    {
        throw "Could not retrieve '$AzureConnectionName' connection asset. Check that you created
        this first in the Automation service."
    }
    # Get the Azure management certificate that is used to connect to this subscription
    $Certificate = Get-AutomationCertificate -Name $AzureConn.AutomationCertificateName
    if ($Certificate -eq $null)
    {
        throw "Could not retrieve '$AzureConn.AutomationCertificateName' certificate asset. Check
        that you created this first in the Automation service."
    }
}
```

```

}
# Set the Azure subscription configuration
Set-AzureSubscription -SubscriptionName $AzureConnectionName -SubscriptionId
$AzureConn.SubscriptionID -Certificate $Certificate
}

```

You would call this runbook at the start of almost any Windows PowerShell Workflow script that is connecting to Azure to be able to work on those resources. By importing it, publishing it, creating global asset objects, and then calling it from another runbook using those assets, you can leverage common code over multiple runbooks and make it much easier to perform common tasks. Following is the code to call the Connect-Azure runbook from another runbook called Connect-AzureVM. Call Connect-AzureVM to set up a connection to an Azure VM. To do this, you first have to import and publish the Connect-Azure runbook.

```

workflow Connect-AzureVM
{
[OutputType([System.Uri])]
Param
(
    [parameter(Mandatory=$true)]
    [String]
    $AzureConnectionName,
    [parameter(Mandatory=$true)]
    [String]
    $ServiceName,
    [parameter(Mandatory=$true)]
    [String]
    $VMName
)
# Call the Connect-Azure runbook to set up the connection to Azure using the Automation connection
asset
    Connect-Azure -AzureConnectionName $AzureConnectionName

InlineScript {
    # Select the Azure subscription we will be working against
    Select-AzureSubscription -SubscriptionName $Using:AzureConnectionName
    # Get the Azure certificate for remoting into this VM
    $winRMCert = (Get-AzureVM -ServiceName $Using:ServiceName -Name $Using:VMName | select
-ExpandProperty vm).DefaultWinRMCertificateThumbprint
    $AzureX509cert = Get-AzureCertificate -ServiceName $Using:ServiceName -Thumbprint
$winRMCert -ThumbprintAlgorithm sha1
    # Add the VM certificate into the LocalMachine
    if ((Test-Path Cert:\LocalMachine\Root\$winRMCert) -eq $false)
    {
        Write-Progress "VM certificate is not in local machine certificate store - adding it"
        $certByteArray = [System.Convert]::fromBase64String($AzureX509cert.Data)
        $CertToImport = New-Object
System.Security.Cryptography.X509Certificates.X509Certificate2 -ArgumentList ($certByteArray)
        $store = New-Object System.Security.Cryptography.X509Certificates.X509Store "Root",
"LocalMachine"
        $store.Open([System.Security.Cryptography.X509Certificates.OpenFlags]::ReadWrite)
        $store.Add($CertToImport)
        $store.Close()
    }
}
}

```

```

    }
    # Return the WinRM Uri so that it can be used to connect to this VM
    Get-AzureWinRMUri -ServiceName $Using:ServiceName -Name $Using:VMName
  }
}

```

The runbook calling order here is the Connect-AzureVM runbook calling the Connect-Azure runbook. If using inline script, the Connect-Azure runbook must be published first, and then the Connect-AzureVM runbook published after. The reason order matters is due to a feature in Azure Automation when using inline script, which is what both of these runbooks do. Any runbook called inside inline script must be published before its parent, the calling runbook. Inline Windows PowerShell script is used to run commands or expressions in a workflow that are valid in Windows PowerShell, but not valid in workflows.

To manage this feature, run the commands in an inlineScript activity. You also can use an inlineScript activity to run Windows PowerShell scripts (.ps1 files) in a workflow. The inlineScript activity runs commands in a standard, nonworkflow Windows PowerShell session and then returns the output to the workflow. It is valid only in workflows. The commands in an inlineScript script block run in a single session and can share data, such as the values of variables. By default, the InlineScript session runs out-of-process; that is, it runs in its own process, not in the workflow process, but you can change this default by changing the value of the OutOfProcessActivity property of the session configuration.

If the publishing doesn't occur in this order, an error message states that it can't find the called runbook. Even though the runbook exists and is published, if it's not published before its parent runbook, it will not be called. This problem can be hard to find.

Using Azure Active Directory without the Connect-Azure runbook

As mentioned in the "Azure Active Directory and automation" section at the start of this chapter, recent updates to Azure Active Directory, Windows PowerShell, and Azure Automation have given the option to authenticate without using certificates in favor of authenticating using Azure AD. Using Azure AD for authentication is the more strategic method than using certificates and is the clear future direction with respect to authentication, not just in Azure Automation, but in almost all Azure services that require that service.

In this case, the calling module (such as Connect-AzureVM) no longer needs to call the Connect-Azure runbook to authenticate. Instead, make calls to Get-AutomationPSCredential and pass in the name of the Azure AD Automation account that the script is running under and that needs to authenticate. This action returns a credential object that is immediately passed in the call to Add-AzureAccount. Here is another code sample of this authentication process to show you how it is done as a recommended best practice.

```

workflow MySampleWorkflow
{

```



```

param
(
    #include your list of parameters
)

# Get the credential to use for Authentication to Azure.
$Cred = Get-AutomationPSCredential -Name 'Azure AD Automation Account'

# Connect to Azure
$AzureAccount = Add-AzureAccount -Credential $Cred

# Begin processing of workflow
}

```

Schedule assets

When you want to execute your runbooks automatically at either a specific date and time or on a recurring basis, you can use a schedule asset. No manual intervention is necessary to start schedule assets. Azure will allocate resources, load, and then execute the runbook when the schedule triggers. When the script completes, Azure will manage the release of execution resources.

Although schedules are assets, they differ slightly from assets such as connections, certificates, and variable assets. The difference is that you never insert or call a schedule from script code. Rather, you will link a runbook to a schedule. A schedule asset determines when runbooks that are linked to it can run. A schedule asset triggers runbook execution when the schedule is activated. You select a published runbook, and on its Schedule tab, you can choose to link to a new schedule. Draft runbooks cannot be linked to a schedule.

Schedule assets are the alternative for manual invocation of runbooks or being called by code from another runbook. Scheduling is just a deeper level of automation beyond just having a script: It's like automating the automation!

Creating a schedule asset

To create a schedule asset, do the following:

1. In the Azure Management Portal, click Automation, select the Azure Automation account, and then click Assets. Click Add Setting.
2. In the Add Setting dialog box, click Add Schedule. On the Configure Schedule page, enter a name that is unique to that Azure Automation account, enter an optional description, and then click the right arrow.
3. In the Configure Schedule dialog box, you can choose to run the schedule one time, hourly, or daily. Depending on the option you select, the remainder of the dialog box entry fields will change slightly. The options are as follows:

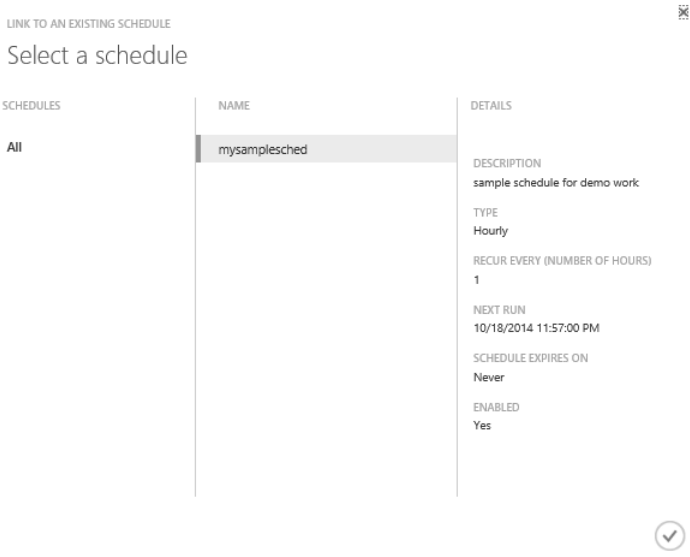
- **One Time** Choose a start date and time.
- **Hourly** Choose a start date and time. As an option, you can select the Set Schedule Expiration Time check box and then enter a date and time to ensure the schedule expires at that date and time. Enter a value in the Recur Every (Number Of Hours) field.
- **Daily** The same settings are available here as in Hourly, but you choose the number of days to recur instead of number of hours.

Configuring the granularity of scheduling options here is not like setting up a meeting in Outlook. You can only run a schedule at a maximum once per hour and you set an expiration time for it to end. For example, if you set the schedule Start Time as 8 pm on September 1 for a recurrence of every two days, and you set the Schedule Expires On date to September 10, the schedule would run on September 1, 3, 5, 7, and 9, but it will not run on September 11 because the expiration date is September 10.

Using the schedule

After you create the schedule, you can link it to a runbook. In the Azure Management Portal, display the list of runbooks for the Azure Automation account, and verify that the runbook that you want to link to a schedule is published. If the runbook isn't published, publish it before you try to link a schedule to it.

In the Azure Management Portal, select the runbook, click Schedule, and then choose to either link to a new schedule or an existing schedule. If you select a new schedule, you can use the Configure Schedule dialog box to create a new schedule. If you select an existing schedule, use the Link To An Existing Schedule dialog box to choose a schedule and display its details.



If the runbook you are linking to the schedule has no parameters that it needs specified, click the check mark to complete the link process. If the runbook does require parameters, however, you will have to specify the runbook parameter values. As shown in Figure 3-9, in the Specify The Runbook Parameter Values dialog box, enter values to pass to the runbook automatically when the schedule invokes. If you manually invoke a runbook that has parameters, you enter the values when the runbook is run. However, the Connect-AzureVM runbook requires you to enter values for all three fields when linking it to a schedule instance.

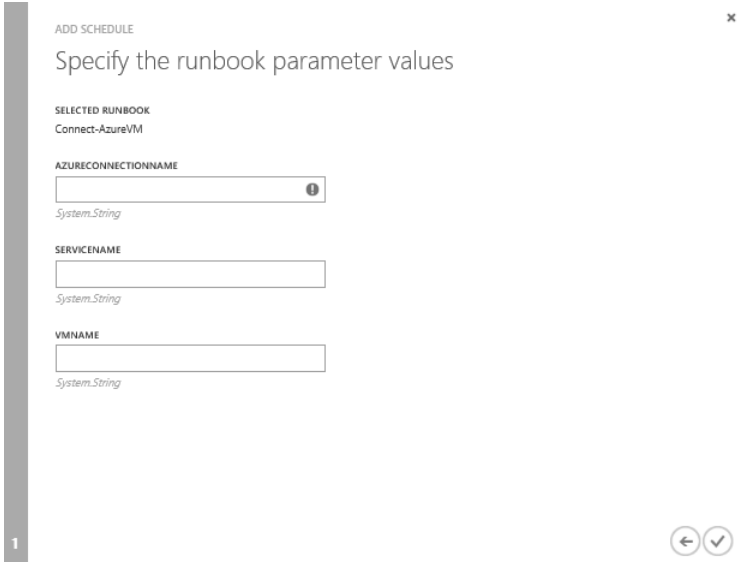


FIGURE 3-9 Enter values for parameters when specifying the runbook parameter values.

About the author



Mike McKeown is a Microsoft Azure MVP who is employed as a Principal Cloud Architect with Aditi Technologies. He spent almost two decades with Microsoft in various roles and has spent over 25 years working within various IT roles. This has given Mike a very unique breadth, as well as depth, of the IT environment from the view of development, management, infrastructure, sales, and the customer. He has experience in the cloud around both the Infrastructure and Platform as a Service solution models. His passion is to help stakeholders or customers define their business/system requirements, and then apply cloud architecture patterns and best practices to meet those goals.

Mike writes white papers for MSDN, blogs about Azure on his blog at www.michaelmckeown.com, develops Azure video training content for Pluralsight, and is a speaker at both regional and national conferences. You can follow his experiences with Azure on Twitter at @nwoekcm.

Mike lives in Charlotte, NC with his wife Tami and five kids Kyle, Brittany, Adrianna, Michael Jr, and Sean. He plays the drums, is active in his church, and loves to work out regularly.




From technical overviews to drilldowns on special topics, get *free* ebooks from Microsoft Press at:

www.microsoftvirtualacademy.com/ebooks

Download your free ebooks in PDF, EPUB, and/or Mobi for Kindle formats.

Look for other great resources at Microsoft Virtual Academy, where you can learn new skills and help advance your career with free Microsoft training delivered by experts.

Microsoft Press



Now that
you've
read the
book...

Tell us what you think!

Was it useful?

Did it teach you what you wanted to learn?

Was there room for improvement?

Let us know at <http://aka.ms/tellpress>

Your feedback goes directly to the staff at Microsoft Press,
and we read every one of your responses. Thanks in advance!



Microsoft