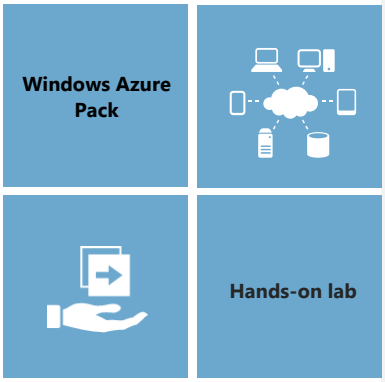
Microsoft

Windows Azure Pack – Introduction to Service Management Automation



Windows Azure Pack (WAP) is a collection of Windows technologies available at no cost that enables the delivery of Windows Azure-style cloud services to external customers and to internal consumers. WAP includes Service Management Automation (SMA), which provides the means to automate the creation, monitoring, and deployment of resources in the WAP environment.

The goal of this lab is to provide a basic understanding of SMA in WAP. You will create Windows PowerShell scripts which use WAP cmdlets that connect to the WAP resources. You will examine the difference between Windows PowerShell scripts and Windows PowerShell workflows. Finally, you will create a runbook that uses a Windows PowerShell workflow.

Produced by HynesITe, Inc
Version 1.0
2/20/2014

holSystems
Learn. Experience. Collaborate.

Introduction

Estimated time to complete this lab

60 minutes

Objectives

After completing this lab, you will be able to:

- Create a Windows PowerShell script that uses some of the Windows Azure Pack cmdlets.
- Create a simple runbook.

Prerequisites

Before working on this lab, you must have:

- A basic understanding of Windows PowerShell, including the use of cmdlets and variables.
- Basic familiarity with the Windows PowerShell ISE console.

Overview of the lab

Windows Azure Pack (WAP) is a collection of Windows technologies available at no cost that enables the delivery of Windows Azure-style cloud services to external customers and to internal consumers. WAP includes Service Management Automation (SMA), which provides the means to automate the creation, monitoring, and deployment of resources in the WAP environment.

The goal of this lab is to provide a basic understanding of SMA in WAP. You will create Windows PowerShell scripts which use WAP cmdlets that connect to the WAP resources. You will examine the difference between Windows PowerShell scripts and Windows PowerShell workflows. Finally, you will create a runbook that uses a Windows PowerShell workflow.

Virtual machine technology

This lab is completed using virtual machines that run on Windows Server 2012 Hyper-V technology. To log on to the virtual machines, press CTRL+ALT+END and enter your logon credentials.

Computers in this lab

This lab uses computers as described in the following table. Before you begin the lab, you must ensure that the virtual machines are started and then log on to the computers.

Virtual Machine	Role
WAPSQL	SQL Server database and Active Directory domain controller for the WAP.Local domain
WAPPortal	Express installation of Windows Azure Pack Portal and Web App Gallery. Configured as an access point for service and tenant admin websites.
WAPSMA	Service Management Automation and Service Bus
WAPSPF	Service Provider Foundation
WAPVMM	System Center Virtual Machine Manager 2012 R2 server
WAPHVS	Hyper-V server (virtualized), file server, and MySQL server
SitesCN	Web Sites Controller

Virtual Machine	Role
SitesFE	Web Sites Front End
SitesMN	Web Sites Management
SitesPB	Web Sites Publisher
SitesWWR	Web Sites Reserved
SitesWWS	Web Sites Shared

All user accounts in this lab use the password **Passw0rd!**

Navigation

Windows Azure Pack uses icons instead of labeled buttons for many tasks. While most are obvious, this document refers to them by action names. Below is a quick reference of action names and corresponding icons.

- Complete, Done or Finished – Check Mark
- Search – Check Mark
- Next – Right Arrow
- Previous – Left Arrow
- Close or Exit – X

Note regarding pre-release software

Portions of this lab may include software that is not yet released, and as such may still contain active or known issues. While every effort has been made to ensure this lab functions as written, unknown or unanticipated results may be encountered as a result of using pre-release software.

Note regarding user account control

Some steps in this lab may be subject to user account control. User account control is a technology which provides additional security to computers by requesting that users confirm actions that require administrative rights. Tasks that generate a user account control confirmation are denoted using a shield icon. If you encounter a shield icon, confirm your action by selecting the appropriate button in the dialog box that is presented.

Note on activation

The virtual machines for these labs may have been built by using software that has not been activated. This is by design in the lab to prevent the redistribution of activated software. The unactivated state of software has been taken into account in the design of the lab. Consequently, the lab is in no way affected by this state. For operating systems other than Windows 8, please press Cancel or Close if prompted by an activation dialog box. If you are prompted by an Activate screen for Windows 8.1, press the Windows key to display the Start screen.

Exercise 1: Understanding Windows Azure Pack Components and Architecture

In this exercise, you will not perform any tasks. You will be provided with an overview of the Windows Azure Pack and how it is deployed in this lab environment. Understanding this environment and deployment is the key to understanding the steps preformed in the lab exercises.

> **Important**: If you already have a good understanding of the Windows Azure Pack and this environment, proceed to Exercise 2.

This lab initially configures the following components:

- Web Sites
- Infrastructure as a Service (IaaS)
- Service Bus
- Automation
- Databases

Windows Azure Pack components: Web Sites

The Windows Azure Pack: Web Sites service uses a minimum of 6 server roles: Controller, Management Server, Front End, Web Worker, File Server, and Publisher. Also required is a server running SQL Server for the Web Sites runtime database. These roles are separate from, and in addition to, the servers that form an express or distributed installation of the Service Management API. The roles can be installed on physical servers or virtual machines.

1. **Web Sites Controller** - The controller provisions and manages the other Web Sites roles. This role is installed first. This role is installed on the SitesCN virtual machine.
2. **Management Server** - This server exposes a REST endpoint that handles management traffic to the Windows Azure Pack Web Sites Management API. This role is installed on the SitesMN virtual machine.
3. **Web Workers** - These are web servers that process client web requests. Web workers are either Shared or Reserved (at minimum, one of each is required) to provide differentiated levels of service to customers. Reserved workers are categorized into small, medium, and large sizes. The SitesWWR is the reserved worker server, and SitesWWS is the shared worker server.

   > **Important**: Because web workers run customer code, they represent a potential risk to the Web Sites infrastructure. After installation, you should configure IP filtering from the Management Portal for Administrators to reduce the risk. For more information, see Configure IP filtering.

4. **Front End** - Accepts web requests from clients, routes requests to web workers, and returns web worker responses to clients. Front end servers are responsible for load balancing and SSL termination. This role is provided by the SitesFE server.
5. **File Server** - Provides file services for hosting web site content. The file server houses all of the application files for every web site that runs on the Web Sites cloud. The file server in this environment is WAPHVS. This is a general purpose server that provides several roles.
6. **Publisher** - Provides content publishing to the Web Sites farm for FTP clients, Visual Studio, and WebMatrix through the Web Deploy and FTP protocols. The publishing server is SitesPB.

In addition, up to three database servers are required. Note that in this environment, all SQL Server databases are located on WAPSQL. There is also a MySQL database instance on WAPHVS.

1. **Service Management API database** - The core installation of the Windows Azure Pack Service Management API uses a SQL Server server to store its configuration data. This database is configured as part of the overall Windows Azure Pack installation and is not specific to Web Sites.
2. **Web Sites runtime database** - Prior to installing Windows Azure Pack: Web Sites, you will need to prepare a SQL Server server to contain the runtime database that Web Sites uses for its operations.
3. **Application databases** - If your usage scenario includes providing database functionality for the tenant web sites, you will need to install separate SQL Server and/or MySQL databases to provide this service.

Windows Azure Pack components: Virtual Machines

Virtual Machines leverage several components to provide tenant based access to virtual machines hosted in a System Center Virtual Machine Manager (VMM) infrastructure.

1. **Service Provider Foundation** - Service Provider Foundation (SPF) is provided with System Center 2012 R2 Orchestrator (and System Center 2012 SP1 - Orchestrator). Service Provider Foundation exposes an extensible OData web service that interacts with Virtual Machine Manager (VMM). This enables service providers and hosters to design and implement multi-tenant self-service portals that integrate IaaS capabilities available on System Center 2012 R2. Service Provider Foundation is installed on WAPSPF.
2. **Virtual Machine Manager** – VMM-specific objects such as clouds, virtual machine templates, hardware profiles, networks, and gallery items are exposed as artifacts which can be used to

create virtual machine services. In this environment VMM is deployed on WAPVMM and the VMM environment has been populated with objects.

Windows Azure Pack components: Databases

You can add one or more Microsoft SQL Server or MySQL server instances for tenants to deploy and use. Tenants also use these databases with the Web Sites service.

Microsoft SQL Server databases can range from simple stand-alone databases to highly available, always-on databases. You should rely on your SQL Server database administrator to install and configure SQL Server for you, ensuing that both Windows and SQL Server authentication are available.

In this environment, SQL Server is deployed on WAPSQL. MySQL is deployed on WAPHVS. Both are very simple installations. The complexity of the SQL implementation is hidden by Windows Azure Pack, so you manage all types the same way, regardless of the underlying infrastructure.

Windows Azure Pack components: Service Bus

Service Bus for Windows Server is a set of installable components that provides the messaging capabilities of Windows Azure Service Bus on Windows Server. Service Bus for Windows Server enables you to build, test, and run loosely-coupled, message-driven applications in self-managed environments and on developer computers.

The purpose of Service Bus for Windows Server is to provide similar capabilities across Windows Azure and Windows Server, and to enable flexibility in developing and deploying applications. It is built on the same architecture as the Service Bus cloud service and provides scale and resiliency capabilities. The programming model, Visual Studio support, and APIs exposed for developing applications are symmetric to that for the cloud service making it easier to develop applications for either, and switch between the two. Going forward, the experience for managing entities on the Windows Azure Management Portal will be consistent across the on-premises and cloud versions.

Service Bus is installed on WAPSPF.

Windows Azure Pack components: Service Management Automation

Service Management Automation is a workflow management solution for Windows Azure Pack for Windows Server. It enables you to automate the creation, monitoring, and deployment of resources in your environment.

Service Management Automation can be used to leverage existing System Center infrastructure and automation, including System Center Orchestrator runbooks.

Service Management Automation is installed on WAPSMA.

Exercise 2: Understand the Use of Windows PowerShell Cmdlets in WAP

Windows Azure Pack ships with a number of Windows PowerShell modules that provide the cmdlets to automate administrative tasks, such as the creation, configuration, and monitoring of resources.

In this exercise, you will examine some of the Windows PowerShell cmdlets that are used to provide automation in the WAP environment. Also, you will learn how to establish a connection to WAP so that you can use the Windows PowerShell cmdlets to create, monitor, and manage resources in WAP.

Ensure Windows PowerShell modules are imported

The WAP components in this lab are installed on separate servers. Because the startup order can vary somewhat in the virtual lab environment, and because the WAP components are somewhat sensitive to startup order, it may be the case that one of the necessary Windows PowerShell modules for this lab will not import properly when required. In this task, you will ensure that the Windows PowerShell module is available for use in the lab.

- Begin this task logged on to **WAPPortal** as **WAP\Administrator** using the password **Passw0rd!**

1. On WAPPORTAL, on the taskbar, right-click **Windows PowerShell**, and then click **Run ISE as Administrator**.
2. In the Commands tab on the right-hand side, in the Modules drop-down list, select **MgmtSvcConfig**.
3. In the results, click **Get-MgmtSvcSetting**, and then click **Show Details**.
4. If the module imports without error, leave the Windows PowerShell ISE console open, and then proceed directly to the next task without performing the subsequent steps in this task.
5. If you get an error message, in the console pane, type **restart-computer**, and then press ENTER.
6. Repeat steps 1 – 4 after the computer has restarted.

Create demonstration Windows PowerShell script to list available plans

To execute a Windows PowerShell cmdlet or script against the WAP environment, you must establish a connection to the WAP environment and present an authentication token. In this task, you will create a simple Windows PowerShell script that will allow you to use the Get-MgmtSvcPlan cmdlet to query the plans that are available in the WAP environment.

- Begin this task logged on to **WAPPortal** as **WAP\Administrator** using the password **Passw0rd!**

1. On WAPPortal, from the taskbar, open **File Explorer**.

2. In File Explorer, navigate to the **DVD** drive, and then copy the LabFiles folder to the root of the **C:** drive so that you have a folder named **C:\LabFiles**.
   - The LabFiles folder contains completed solutions for the scripts you use in this lab. Some of the scripts presented in this lab are fairly lengthy, and you may wish to use the supplied solutions as an alternative to typing the scripts.
3. In the Windows PowerShell ISE console, click **View**, and then click **Show Script Pane**.
4. In the Untitled1.ps1 tab, enter the following lines of code.
   - This code can be found in the **c:\solutions\displayplans.ps1** script file.

```
# Set environment-specific variables
$WAPServer="WAPPortal.wap.local"
#Set connection variables and retrieve token from Authentication site for logged-on user
$AdminURI = "https://" + $Using:WAPServer + ":30004"
$AuthSite = "https://" + $Using:WAPServer + ":30072"
$ClientRealm = "http://azureservices/AdminSite"
$token = Get-MgmtSvcToken -Type Windows -AuthenticationSite $AuthSite -ClientRealm $ClientRealm –DisableCertificateValidation
#Retrieve and display plan information
$Plan = Get-MgmtSvcPlan -AdminUri $AdminURI -Token $token
Return $Plan
```

   - In this script, you are creating variables that contain the connection information that is required to connect to WAP. You are then using the values in the variables to execute the cmdlet to get the plan data.
   - The default port specified for the AdminURI endpoint to access the REST APIs must be 30004. The default port number for the Admin authentication site endpoint is 30072.
   - The Get-MgmtSvcToken cmdlet gets a security identity token from a trusted Security Token Service (STS). In this case, the token is retrieved from the Admin Authentication site using the logged-on credentials of the authorized user. The token is presented to the Admin API. For a number of WAP cmdlets, you must acquire a token before cmdlets, such as Get-MgmtSvcPlan, can be used. You must also pass http://azureservices/AdminSsite as the client realm parameter in this cmdlet. For more information on claims-based authentication in WAP, please consult the Whitepaper, Claims-Based Identity in Windows Azure Pack.
5. On the toolbar, click **Run Script** ( ).
   - Only one plan is displayed in the output.

- In WAP, a plan contains one or more services (Web Sites, SQL Database, Virtual Machines, Service Bus, etc.) and service quotas for each service. Subscriptions, add-ons, and resource providers are linked to plans.

6. In the Windows PowerShell output pane, type the following command, and then press ENTER.

```
$Plan | Select –ExpandProperty ServiceQuotas | FL
```

   - The properties of the plan include objects that have additional properties. If you wish to view these properties, you need to expand the properties of the object as in the above command. Note that the services associated with the plan and their quota settings (another nested object) are displayed in the output.

7. In the Windows PowerShell ISE console, click **File**, click **Save As**, and then save the script as **C:\LabFiles\DisplayPlans.ps1**.
8. Leave the Windows PowerShell ISE console and the DisplayPlans.ps1 script open for the next task.

Create a WAP tenant user in the WAP portal

In the previous task, you learned how to retrieve information about a WAP resource using Windows PowerShell cmdlets. The Windows PowerShell cmdlets can also be used to create objects. In this task, you will create two scripts to create a user object. The first script is a demonstration script to show, in general, how to create an object, in this case a tenant user account. Although the output of the first script will appear to indicate success, you will not be able to log in as the newly created user. The reason is that creating user objects is a two-step process that involves creating the user object at the Service Management API layer, and then integrating the user into an authentication provider.

In this task, you will create a script that demonstrates how to create resource objects in WAP. You will then verify that the user is created in the portal.

- Begin this task logged on to **WAPPortal** as **WAP\Administrator** using the password **Passw0rd!**

1. In the Windows PowerShell ISE console, in the DisplayPlans.ps1 tab, click the mouse anywhere in the script window, press CTRL-A, and then press CTRL-C to copy the entire script to the clipboard.
2. Click **File**, and then click **New**.
3. In the Untitled2.ps1 tab, click the mouse anywhere in the script window, and then press CTRL-C to copy the script from the clipboard.
4. Locate the comment **#Retrieve and display plan information**.
5. Delete the comment and the lines of code that follow. The lines to delete are shown below.

```
#Retrieve and display plan information
$Plan = Get-MgmtSvcPlan -AdminUri $AdminURI -Token $token
Return $Plan
```

6. At the top of the script, under the section that sets the $WAPServer variable, add the following lines.

```
# Set User Specific Variables
$UserEmail = “AliceCiccu@contoso.com”
$UserName = “Alice Ciccu”
$Password = "Passw0rd!"
$PasswordQ = "DemoQ"
$PasswordA = "DemoA"
```

7. At the bottom of the script, add the following lines.

```
#Create User at the Service Management API layer (within WAP)
$User = Add-MgmtSvcUser -Email $UserEmail -Name $UserName -State
Active -ActivationSyncState InSync -AdminUri $AdminURI -Token $token
```

8. The script should look this on completion:

```
#Set Environment-specific variables
$WAPServer= "WAPportal.wap.local"

#Set User Specific Variables
$UserEmail = "AliceCiccu@contoso.com"
$UserName = "Alice Ciccu"
$Password = "Passw0rd!"
$PasswordQ = "DemoQ"
$PasswordA = "DemoA"

#Set connection variables and retrieve token from Authentication site for logged on user

$AdminURI = "https://" + $WAPServer + ":30004"
$AuthSite = "https://" + $WAPServer + ":30072"
$ClientRealm = "http://azureservices/AdminSite"
$token = Get-MgmtSvcToken -Type Windows -AuthenticationSite $AuthSite -ClientRealm $ClientRealm -DisableCertificateValidation

#Create User at the Service Management API layer (within WAP)
$User = Add-MgmtSvcUser -Email $UserEmail -Name $UserName -State Active -ActivationSyncState InSync -AdminUri $AdminURI -Token $token
```

9. Click **File**, click **Save As**, and then save the script as **C:\LabFiles\AddUser.ps1**.
10. On the toolbar, click **Run Script** ( ).
    - There is no output from the script.
11. From the taskbar, open **Internet Explorer**.
    - **CAUTION:** Do not close the Windows PowerShell ISE console.
12. On the favorites bar, click **WAP – Admin**.
13. In the Windows Security dialog box, log on as **Administrator** using the password **Passw0rd!**
14. In the Service Management Portal, in the left navigation pane, click **USER ACCOUNTS**.
    - The account created by the script is visible, along with an account for admin@contoso.com.
15. In Internet Explorer, open a new tab.
16. On the Favorites bar, click **WAP-Tenant**.
17. On the Login screen, type **AliceCiccu@contoso.com** as the email address and **Passw0rd!** as the password, and then click **submit**.

- The log in fails because the user object only exists at the Service Management API layer. The failed log in is an expected result.

Integrate user account with authentication provider

In WAP, the authentication and authorization processes are separate. This allows flexibility to plug in custom authentication systems, for example, Active Directory Federation Services (AD FS). For more information on this topic, please see http://blogs.technet.com/b/privatecloud/archive/2014/02/03/creating-users-in-windows-azure-pack.aspx.

In the previous task, you added a user account to the WAP portal. However, because the account had not been integrated with the tenant authentication site, the user was not able to log in. In this task, you will add code to the script you created in the previous task to integrate the user into the tenant authentication site.

- Begin this task logged on to **WAPPortal** as **WAP\Administrator** using the password **Passw0rd!**
  - All code used in this section can be found in the **c:\LabFiles\Solutions\AddUser.ps1** file.

1. In the AddUser.ps1 tab, at the bottom of the script, add the following lines.

```
# Establish ASP Provider Configuration
Add-Type -Path
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\System.Web.dll
$config = New-Object
System.Collections.Specialized.NameValueCollection
```

   - To add users to the membership database, you need to use the ASP.NET membership API. The above code establishes the configuration and is an initial step for creating the provider. In a production environment, you would likely use an ASP.NET application to perform the task of integrating the user account into the tenant authentication site.

2. At the bottom of the script, add the following lines.

```
# Get Connection String for WindowsAuthSite
$connectionString = (Get-MgmtSvcSetting WindowsAuthSite
ApplicationServicesConnectionString).Value
```

   - You need to get a connection string for the tenant authentication site (SQL database) and place the string in a variable.

3. At the bottom of the script, add the following lines.

```
# Configure Provider Settings
$config.Add('enablePasswordRetrieval','false')
$config.Add('enablePasswordReset','true')
$config.Add('requiresQuestionAndAnswer','false')
$config.Add('passwordFormat','Clear')
```

```
$config.Add('requiresUniqueEmail','false')
$config.Add('maxInvalidPasswordAttempts','5')
$config.Add('minRequiredPasswordLength','8')
$config.Add('minRequiredNonalphanumericCharacters','0')
$config.Add('passwordAttemptWindow','30')
$config.Add('applicationName','/')
$config.Add('connectionString', $connectionString)
```

📌 In this example, you are not encrypting the passwords that are inserted into the SQL database. In a production environment, you would likely want to encrypt the passwords using SHA-256 encryption.

4. At the bottom of the script, insert the following lines.

```
# Create ASP Provider
$provider = New-Object System.Web.Security.SqlMembershipProvider
$provider.Initialize('AspNetSqlMembershipProvider', $config)
```

5. After this section, add the following lines to complete the script.

```
# Create User within ASP Provider
$status = 0
$provider.CreateUser($User.Name, $Password, $User.Email, $PasswordQ,
$PasswordA, $true, $null, [ref] $status)
$status
```

6. The newly added section of code will look like this:

```
# Establish ASP Provider Configuration
Add-Type -Path C:\Windows\Microsoft.NET\Framework64\v4.0.30319\System.Web.dll
$config = New-Object System.Collections.Specialized.NameValueCollection

# Get Connection String for WindowsAuthSite
$connectionString = (Get-MgmtSvcSetting WindowsAuthSite ApplicationServicesConnectionString).Value

# Configure Provider Settings
$config.Add('enablePasswordRetrieval','false')
$config.Add('enablePasswordReset','true')
$config.Add('requiresQuestionAndAnswer','false')
$config.Add('passwordFormat','Clear')
$config.Add('requiresUniqueEmail','false')
$config.Add('maxInvalidPasswordAttempts','5')
$config.Add('minRequiredPasswordLength','8')
$config.Add('minRequiredNonalphanumericCharacters','0')
$config.Add('passwordAttemptWindow','30')
$config.Add('applicationName','/')
$config.Add('connectionString', $connectionString)

# Create ASP Provider
$provider = New-Object System.Web.Security.SqlMembershipProvider
$provider.Initialize('AspNetSqlMembershipProvider', $config)

# Create User within ASP Provider
$status = 0
$provider.CreateUser($User.Name, $Password, $User.Email, $PasswordQ, $PasswordA, $true, $null, [ref] $status)
$status
```

📌 This shows only a partial view of the entire script.

7. Click **File**, and then click **Save**.

8. In the script pane, using the mouse, select all the code you entered in this task (shown above), right-click the selected code, and then click **Run Selection**.
   - You have already created the user account, and the variables that you used for the user account creation are still present in the session.
   - **IMPORTANT:** Please ensure you do not select any of the code that you ran in the previous task.
9. Leave the Windows PowerShell ISE console open for subsequent tasks.

Exercise 3: Understand and Create SMA Runbooks

Service Management Automation (SMA) is a set of tools that is integrated as the automation extension in Windows Azure Pack (WAP) for Windows Server. Administrators and developers can leverage the automation extension to create, run, and manage runbooks. SMA runbooks, like their System Center 2012 Orchestrator counterparts, can be used to integrate, orchestrate, and manage IT processes.

SMA runbooks run on the Windows PowerShell Workflow engine. Windows PowerShell Workflow is included with Windows PowerShell 3.0. This means that it is available by default on Windows Server 2012, Windows Server 2012 R2, and Windows 8. Windows PowerShell workflow is also available for down-level installations of Windows Server 2008, Windows Server 2008 R2, and Windows 7 via Windows Management Framework 3.0.

When a runbook is started from either the Service Management Portal or the Start-SmaRunbook cmdlet, the Service Management web service writes the start request to the Automation database. One of the Automation Worker servers retrieves the request and processes the request. Because the Automation Worker server may require access to remote computers or resources, the Windows PowerShell cmdlets in the runbooks need to be able to remotely access those resources. Alternatively, the runbook needs to include the InlineScript command to use Windows PowerShell remoting to run the commands locally on the target computer.

It is important to note that only the computer that runs the workflow (the host) needs to have the WPSW engine installed. The target (managed node) does not need to meet this requirement. You can use WIM or CIM commands on the target node. The use of PowerShell cmdlets on the target node does, however, require that PowerShell 2.0 be installed. A final requirement is that PowerShell remoting be enabled on both the host and target computers.

A workflow is a set of related activities. An activity is a step within a workflows that performs a defined task, such as getting a list of WAP plans or listing virtual machines. Workflows are ideally suited for executing commands that take an extended period to run, need to run in parallel for increased efficiencies, need to survive reboots and disconnected sessions, need to be suspended and resumed without loss of data, and/or need to be throttled or connection-pooled in large-scale or high-availability environments.

Windows PowerShell Workflow allows IT administrators and developers to create workflows using Windows PowerShell syntax, rather than using Extensible Application Markup Language (XAML). This means that current Windows PowerShell scripting knowledge and Windows PowerShell scripts themselves can be leveraged to create workflows. Current XAML-based workflows work in Windows PowerShell Remoting.

In this exercise, you will familiarize yourself with the SMA capabilities in WAP. You will then create a runbook that leverages a sample runbook available in the WAP portal.

Examine SMA capabilities

Service Management Automation (SMA) is a workflow management solution available in Windows Azure Pack that enables you to automate the creation, deployment, and monitoring of resources. In this task, you will examine the capabilities of SMA in the Service Administration portal and look briefly at some of the syntactical conventions used by Windows PowerShell Workflows.

- Begin this task on **WAPPortal** logged on as **WAP\Administrator** using the password **Passw0rd!**

1. Ensure Internet Explorer is open and that you are viewing the WAP Administration portal.
2. In the left navigation bar, click **AUTOMATION**.
   - The dashboard displays the status, both current and historical, of the runbook jobs. Because there has been no runbook activity in this lab environment, no data is displayed in the chart. The quick glance section displays summary information about the number of runbooks, Windows PowerShell modules, and settings (the number of pre-configured variable, connections, credentials, certificates, and schedules).
3. Above the chart and to the right, click the down arrow to the right of **7 days**.
   - You can specify a date range for the information presented in the graph.
4. On the automation page, click **RUNBOOKS**.
   - From this page, you can start, export, import, or delete runbooks.
   - The SMA runbook toolkit (SMART) extends the importing and exporting capabilities of SMA, allowing you to more easily reuse runbooks created in different SMA environments. For more information on SMART, please see http://blogs.technet.com/b/privatecloud/archive/2013/10/23/automation-service-management-automation-sma-runbook-toolkit-spotlight-smart-for-runbook-import-and-export.aspx.
5. To the right of JOB STATUS, click the drop-down box to the reveal the job status filter choices.
   - You can apply a filter to quickly find runbooks according to their job status and dates of the jobs.
6. Click the drop-down box again to collapse the list, and then under NAME, click **Sample-Using-Checkpoints**.
   - Windows Azure Pack provides a number of sample runbooks that can be used as a starting point to introduce additional automation into your environment.
7. On the sample-using-checkpoints page, click **AUTHOR**.
   - The sample runbooks comprise, for the most part, short tutorials on the use of workflow constructs and features. This particular sample shows how you can create a workflow that uses a checkpoints.
8. In the sample-using-checkpoints script, locate the string **workflow Sample-Using-Checkpoints**.
   - "Workflow" indicates the start of the workflow. "Sample-Using-Checkpoints" is the name of the workflow. The workflow itself is contained between the curly braces.

9. Click the back arrow.
10. Click **Sample-Using-Connections**, and then click **AUTHOR**.
    - 📌 **TIP:** Sample-Using-Connections should be just below Sample-Using-Checkpoints
11. On the sample-using-connections page, locate the string **InlineScript**.
    - 📌 "InlineScript is another construct that is specific to workflows. The InlineScript {} block contains a Windows PowerShell script. Windows Workflow Foundation invokes Windows PowerShell to process the script block as a traditional Windows PowerShell script.
12. On the sample-using-connections page, locate the string **–PSComputerName**.
    - 📌 The script block will be executed on a remote computer (the target or managed node) that is indicated by the –PSComputerName parameter. Because of the way that workflows handle computer names, you need to use the –PSComputerName parameter, rather than the –ComputerName parameter you would use in a Windows PowerShell script.
13. Click the back arrow.
14. Click **Sample-Using-Credentials**, and then click **AUTHOR**.
    - 📌 **TIP:** Sample-Using-Credentials should be just above Sample-Using-Connections.
15. On the sample-using-credentials page, locate the **param** keyword.
    - 📌 Parameters can be added to workflow. In this case, the server name is being added as a parameter. As with functions, you can define default values for parameters.
16. On the sample-using-credentials page, in the InlineScript block, locate the string **$Using:ServerName**.
    - 📌 The $ServerName variable defined at the top of the script in the parameter block needs to be passed to the InlineScript block that will run in a Windows PowerShell remote session on a computer defined by the –PSComputerName variable. In Windows PowerShell 3.0, the **$using:** scope modifier allows the variable to be passed to the remote session.
17. On the sample-using-credentials page, click **SCHEDULE**.
    - 📌 You can schedule runbooks.
18. On the sample-using-credentials page, click **CONFIGURE**, and then spend a few moments examining the configuration settings.
19. On the portal taskbar, click **NEW**.
20. Click **RUNBOOK**, and then click **QUICK CREATE**.
21. In RUNBOOK NAME, type **Get-TestPlan**.

- A recommended best practice is to name runbooks using the same verb-noun conventions used for Windows PowerShell cmdlets. For information on Windows PowerShell naming conventions, see http://msdn.microsoft.com/en-us/library/ms714428(VS.85).aspx.

22. In DESCRIPTION, type **Display Plans**.
23. In TAGS, type **Plan**.
    - Tags are metadata that you can add to the runbook. They are free-formed, allowing flexibility in determining what is appropriate.
24. Click **CREATE**.
25. Click **OK**.
26. Click **Previous** (the left arrow).
27. On the automation page, click **Get-TestPlan**.
    - **TIP**: You may need to refresh your view to see the new plan.
28. On the get-testplan page, click **AUTHOR**.
29. Click **DRAFT**.
30. Place the mouse after the opening brace on line 2, and then click **INSERT**.
    - You can insert other runbooks, settings (discussed later in this lab), and activities.
31. Click **ACTIVITY**.
32. On the INSERT ACTIVITY page, click **CIMCMDLETS**, click **Get-CimInstance**, and then click **Next** (right arrow).
33. On the Parameters for 'Get-CimInstance' page, in the PARAMETER SET drop down, select **QueryComputerSet**, and then click **Cancel** (x).
    - This is just a demonstration to show how runbooks can be created in the WAP portal directly. You are not going to create a Quick Create runbook.
34. On the Get-TestPlan page, click **MANAGE**.
    - You can import additional Windows PowerShell modules if you require them to run on the WAPPortal or if you want to use them to create runbooks using the Service Management portal.
35. Click the back arrow.
36. If prompted, click **OK** to acknowledge that all changes will be lost.
37. On the automation page, ensure Get-TestPlan is selected, click **DELETE**, and then click **YES**.
    - You will create runbooks in subsequent tasks.
38. On the automation page, click **ASSETS**.
    - The page lists the modules that are currently installed.

39. On the Service Manager Portal taskbar, click **ADD SETTING**.
    - 📌 Settings are reusable configuration objects that can be used in runbooks. For example, using Settings you could configurable a reusable connection to Virtual Machine Manager that you use in runbooks.
40. Click **ADD CONNECTION**.
41. In the CONNECTION TYPE drop-down list, select **MgmtSvcAdmin**.
42. In NAME type **MgmtSvcAdminConnection**.
    - ⚠ **IMPORTANT**: Please type this name exactly as written. You will use this connection setting in a subsequent task.
43. Click **Next** (right arrow).
44. On the Configure connection properties page, in PASSWORD, type **Passw0rd!**
45. In COMPUTERNAME, type **wapportal.wap.local**.
46. In USERNAME, type **WAP\Administrator**, and then click **Done** (check mark).
    - 📌 The connection appears in the list of assets.
47. Leave the Service Management administrative portal open for the next task.

Create a runbook from a sample

In this task, you will export a sample runbook, modify it for the lab environment, and then import and run the modified runbook.

✎ Begin this task logged on to **WAPPortal** as **WAP\Administrator** using the password **Passw0rd!**

1. In the Service Management portal web page you left open from the previous task, in the Automation node, click **RUNBOOKS**.
2. Select **Sample-Managing-Plans**.
   - 📌 Make sure you select a cell to the right of the name. If you do select the name and cause the sample-managing-plan page to appear, click Previous.
3. Click **EXPORT**, and then click **YES**.
4. When prompted to open or save the file, click the drop-down to the right of Save, and then click **Save as**.
5. In the Save As dialog box, in File name, type **C:\LabFiles\New-SQLPlan.ps1**, and then click **Save**.
   - 📌 A warning appears that tells you that the signature is corrupt or invalid. This is expected.
6. Click the **x** beside View downloads to dismiss the warning.
7. Switch to the Windows PowerShell ISE console you left open in a previous exercise.

8. In the Windows PowerShell ISE console, click **File**, click **Open**, and then open **C:\LabFiles\New-SQLPlan.ps1**.
9. Locate the string **workflow Sample-Managing-Plans**, and then change it to **workflow New-SQLPlan**.
   - It is necessary to use a new name for this this runbook. When you import in a subsequent step, you will be creating a new runbook that must have a unique name. As mentioned earlier, it is a good idea to use the Windows PowerShell naming conventions.
10. Locate the **param** keyword and spend a few moments examining the code enclosed in parentheses.
    - Two parameters will be created for this runbook. The first one is mandatory and requires that a plan display name be provided when the runbook is executed. The second is not mandatory but provides a default value for the plan quota setting if one is not provided in the UI.
11. Locate the **InlineScript** block and spend a few moments examining the code.
    - You may notice that, instead of using hardcoded values for the connection data, this Windows PowerShell script derives connection data values using a number of cmdlets. This makes the script portable without modification.
    - Note that the InlineScript uses Windows PowerShell remoting to execute on WAPPortal.wap.local.
12. At the bottom of the InlineScript block, locate the line beginning with the **Update-MgmtSvcPlanQuota** cmdlet.
    - **TIP**: This will be approximately line 80.
13. In this line, replace **$using:AuthenticationSiteURL** with **–AdminUri $AdminSiteURL**. The line should look like the following:

```
Update-MgmtSvcPlanQuota -AdminUri $AdminSiteURL -token $token -DisableCertificateValidation -PlanId $plan.Id -QuotaList $QuotaList
```

    - It is necessary to correct the syntax of this line so that the workflow will run without error.
14. After the last brace, locate the comment **#SIG # Begin signature block**.
15. Delete this line and all the remaining lines that follow it to the bottom of the script.
    - The runbook in this lab environment cannot use the digital signature that is present here.
16. Click **File**, and then click **Save**.
17. Switch to the Service Management admin portal in Internet Explorer that you left open in the previous task.
18. On the automation page, click **IMPORT**.
19. Click **BROWSE FOR FILE**.

20. In the Choose File to Upload dialog box, select **C:\LabFiles\New-SQLPlan.ps1**, and then click **Open**.
21. Click **Done** (check mark).
22. On the automation page, click **New-SQLPlan**.
    - The start icon at the bottom of the page is greyed out. The runbook needs to be published before it can execute.
23. On the New-SQLPlan page, click **AUTHOR**.
24. Click **DRAFT**.
25. On the portal taskbar, click **PUBLISH**, and then click **YES**.
26. Click **DASHBOARD**.
27. On the portal taskbar, click **START**.
28. In PLANDISPLAYNAME, type **SQL-Plan1**, leave the SQLQUOTASETTINGVALUE blank, then and click **Done**.
    - If you want to enter a SQL Quota Setting so that the plan uses something other than the default values, you need to enter the string exactly as follows, changing only the values:
      [{"displayName":"Default","groupName":"Default","resourceCount":"10","resourceSize":"1024","resourceSizeLimit":"1024","offerEditionId":"081313063701","groupType":null}].
      You can copy and paste this string from the .ps1 file you imported; however, please ensure you remove the leading and trailing single quotes.
29. Wait for the runbook job notification area to disappear, and then click **JOBS**.
30. Ensure that STATUS is **Completed**, and then click the date and time entry in the JOB START cell.
31. On the DASHBOARD page, scroll down to see the output of the job.
32. Click **HISTORY**.
    - You should see three outputs listed for the history. If there were a problem with the runbook, one or more of these outputs would provide a visual indication where the problem resides for troubleshooting purposes.
33. Click the topmost **Output**, and then click **VIEW DETAILS**.
    - If there were a problem, you would see the error message that was returned as part of the output.
34. Click **Done**.
35. Click **VIEW SOURCE**.
36. Click **Done**.
37. In the left navigation, click the **Plans** node ().
    - After a few moments, the newly added SQL plan should appear. You may have to refresh the page.

38. On the plans page, click **SQL-Plan1**.
39. Under plan services, click **SQL Servers**.
    - The settings used in the parameter in the runbook are displayed.