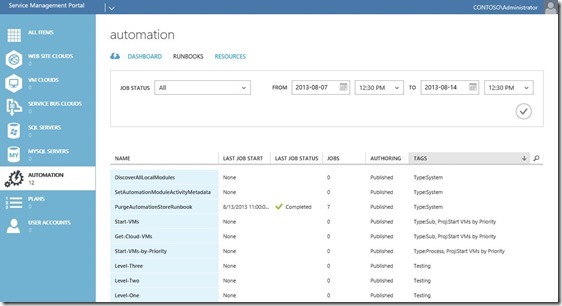
Hello Readers!  Well, it has been a while since I’ve posted, but it has been for a very good reason.  I have something exciting to share that I believe will be of great benefit to those of you embarking down the path of Service Management Automation (SMA) within the Windows Azure Pack (WAP) and not only have the desire to import Runbooks created by other Runbook authors, but also wanting to share Runbooks you’ve written in a streamlined fashion.

## <http://aka.ms/IntroToSMA>



# What’s the Importance of this Post for SMA?

Well, this post introduces SMART (SMA Runbook Toolkit) for Runbook Import and Export.  What is so important about this solution? Well read on but in a nutshell, this solution will allow you to programmatically export all SMA Runbooks from your SMA environment into an atomic XML file representing each Runbook and dependencies such as variables, schedules,etc.  Using this same solution, you can take those atomic XML Runbook definitions and import them into the same or different SMA environments keeping most all the configuration data in tact.  Imagine having the option to backup your blessed copy of SMA Runbooks on a nightly basis or use this solution to move Runbooks from DEV –> QA –> PROD environments all from PowerShell. Curious now? Good, let’s get started! Open-mouthed smile

Click the Download Button to Get SMART for Runbook Import and Export

[BC-DLButtonDark](http://gallery.technet.microsoft.com/SMART-SMA-Runbook-Toolkit-2ba4f4c5)

UPDATE

Please see the following post to review some recent updates to this solution.

[**Automation–SMA Runbook Toolkit (SMART) for Runbook Import / Export–Updated!**](http://blogs.technet.com/b/privatecloud/archive/2014/03/10/automation-sma-runbook-toolkit-smart-for-runbook-import-export-updated.aspx)

The specific updates for this release

* Ability to capture encrypted values from variables and credentials and store them within the XML export file (if run from inside SMA)
* Ability to create your original credentials with this captured data in the target SMA environment you import them into
* Minor bug updates

The updates are available in the above “DOWNLOAD” linked version.

# Background

First, let’s start with a little background on this subject.  SMA ships with a great PowerShell provider that gets you access to specific pieces within SMA related to Runbooks.

* **Get-SMARunbook** – get description, log settings, tags, schedules (see below for more details)
* **Get-SMARunbookDefinition –** used to get details of definition content explained a bit below such as the PowerShell
* **Edit-SMARunbook –** allows you to edit an existing Runbook by updating with another PS1
* **Import-SMARunbook** – allows you to import a Runbook (PS Workflow)
* **Publish-SMARunbook** – allows you to publish a Runbook by name

The above are a handful of cmdlets available to get certain details about a Runbook or even a series of Runbooks  just by using PowerShell.  This gets us to the “front door” so to speak, but we need to be able to get the full picture of what encompasses a Runbook (and all its dependencies) or even a Project and all dependent artifacts that encompass that project.  Let’s start with an overview of the components of an SMA Runbook / SMA Project.

# What Am I Covering in This Post?

**Components of a Runbook Project** – details on all artifacts are involved in a Runbook project end to end.

**Solution Breakdown** – what is SMART and how do you get it installed and start leveraging it

* What is in Scope for SMART for Import and Export
* What is out of Scope for SMART for Import and Export
* Getting things setup to start using SMART

**Examples leveraging the solution –** give you some execution worthy script snippets you can leverage to run the Import / Export process in your SMA environment.

**UPDATE     Thank you PowerShell MVP**[Aleksandar Nikolić](http://blogs.technet.com/106690/ProfileUrlRedirect.ashx)**for you input and suggestions on noun-verb as well as switching from “switches” to “Boolean” for all PowerShell.  Makes good sense and was happy to roll it into the solution as improvements.  Keep the feedback coming!**

# Components of a Runbook Project

To give you some background on why this solution was made, I wanted to elaborate a bit on what makes up a Runbook and its dependencies.

### Runbook PowerShell

The Runbook PowerShell itself is a configuration that can be gathered via **Get-SMARunbookDefinition** and there are two possible definitions (Draft and Published)   
Example

|  |  |
| --- | --- |
| 001  002 | $RunbookDefinition = Get-SMARunbookDefinition –name “Runbook-Name” –Type “Published” –WebServiceEndpoint “https://localhost”   $RunbookContent = $RunbookDefinition.Content #this actually retrieves the Runbook PowerShell defined |

### Description

Each Runbook has a field for description. This can be used to define the function of the Runbook beyond what is implied by its name. These are defined once for both Published and Drafted versions on the **Configuration** tab of each Runbook.

Example

|  |  |
| --- | --- |
| 001  002 | $Runbook = Get-SMARunbook –name “Runbook-Name” –WebServiceEndPoint “https://localhost”   $RunbookDescription = $Runbook.Description #gets you the description value |

### Tags

Tags are applied to each Runbook to help you organize your Runbooks within WAP.  These allow you to sort and search according to certain data such as “Proj:SMART”.   
Example

|  |  |
| --- | --- |
| 001 | $Runbook.Tags #will get you the tags defined for your Runbook |

### Log Settings

Each Runbook has three defined log configurations that can be assigned [Log Debug, Log Progress, Log Verbose] and they are all set on the **Configuration** tab of the Runbook..   
Example

|  |  |
| --- | --- |
| 001  002  003 | $Runbook.LogVerbose  $Runbook.LogDebug  $Runbook.LogProgress |

### Schedules

Schedules are defined independently and also directly associated with Runbooks.  This makes it easy to find out what schedules are leveraged by the Runbook when you want to export for sharing.   
Example

|  |  |
| --- | --- |
| 001 | $Runbook.Schedules #gets you the list of schedules you can then also assign to a variable to get each schedule independently |

### SMA Variables

SMA Variables are leveraged within an SMA Runbook by calling the cmdlet **Get-SMAVariable** and unfortunately have no direct dependency mapping to the Runbook other than being named directly within the Runbook PowerShell script itself.   
Example

|  |  |
| --- | --- |
| 001 | $SERVER = Get-SMAVariable –name ‘ServerName’ –WebServiceEndpoint “https://localhost” |

### SMA Credentials

SMA Credentials are leveraged within an SMA Runbook by calling the cmdlet **Get-AutomationPSCredential** and also unfortunately have no direct dependency mapping to a particular Runbook other than being named directly within the Runbook PowerShell script itself.   
Example

|  |  |
| --- | --- |
| 001 | $RunbookCredential = Get-AutomationPSCredential –name ‘Credential’ –WebServiceEndpoint “https://localhost” |

And so you don’t have to do a lot of copy / pasting to play around with some of the existing PowerShell for Runbooks in SMA, here is all of the above in one script block for convenience.

|  |  |
| --- | --- |
| 001  002  003  004  005  006  007  008  009  010  011  012  013 | $RunbookDefinition = Get-SMARunbookDefinition –name “Runbook-Name” –Type “Published” –WebServiceEndpoint “https://localhost”   $RunbookContent = $RunbookDefinition.Content #this actually retrieves the Runbook PowerShell defined   $Runbook = Get-SMARunbook –name “Runbook-Name” –WebServiceEndPoint “https://localhost”   $RunbookDescription = $Runbook.Description #gets you the description value   $Runbook.Tags #will get you the tags defined for your Runbook   $Runbook.LogVerbose   $Runbook.LogDebug   $Runbook.LogProgress    $Runbook.Schedules #gets you the list of schedules you can then also assign to a variable to get each schedule independently |

### Parent and Child Runbook Relationships

As part of having a project that is defined by a top level process Runbook and a series of subroutine Runbooks, there is actually no way to draw the dependency for Parent / Child very easily.  Looking at the example below, you can see this is purely referential and there is no way to know that the Runbook **ChildRunbook** is being called by the parent Runbook **ParentRunbook** or if this call is a cmdlet, function, etc.  The association happens on publish of the parent Runbook and if the child Runbook doesn’t exist, the association doesn’t happen, more on that later.   
Example Parent Runbook

|  |  |
| --- | --- |
| 001  002  003  004 | Workflow ParentRunbook  {          ChildRunbook  } |

Example Child Runbook

|  |  |
| --- | --- |
| 001  002  003  004 | Workflow ChildRunbook  {         Some PowerShell……  } |

So you get the idea, a lot of components go into your Runbook projects but no clear way of extrapolating the data into an atomic “blob” of goodness that can be shared between environments easily (well not without going through what this solution does – so why not use this instead of writing your own Winking smile).

# Solution Breakdown

Ok, so now that you have some background on some of the drivers of this toolkit, let’s talk about what it does and how you can leverage it.

## Introducing SMART (SMA Runbook Toolkit) for Runbook Import and Export

So what is **SMART for Runbook Import and Export**?  Well, basically the premise around this solution is to provide a mechanism to point at your SMA environment, collect up all Runbooks and dependencies, and export those into a series of “atomic” XML blobs that can be then used by the import piece of this solution to bring into another SMA environment.

Click the Download Button to Get SMART for Runbook Import and Export

[BC-DLButtonDark](http://gallery.technet.microsoft.com/SMART-SMA-Runbook-Toolkit-2ba4f4c5)

## In Scope for SMART for Runbook Import and Export

* Runbooks, configuration data, and most assets can be exported to an XML (see below for more details)
* XML created by the SMART export process can import Runbooks, configuration data, and most assets (see below for more details)

### **Runbook Export**

Runbooks can be exported with the following options

|  |  |
| --- | --- |
| **COMPONENT** | **DESCRIPTION** |
| Published and Draft content | Both **Draft** and **Published** **versions** are supported for export and even better they are automatically picked up during export if Runbooks differ in content. |
| Runbook Tags | Runbook tags are exported by default for all Runbooks if present. |
| Runbook Description | Runbook **description** is exported **by** **default** for all Runbooks if present |
| Runbook Logging Options | Runbook **logging preferences** are exported **by default** for all Runbooks if present |
| Runbook Schedules | Runbook **schedules** (Daily only)  are exported using the  **–ExportSchedules $True** or **–ExportAssets $True** to export all assets supported |
| Leveraged SMA Variables | The following variables are supported with some restrictions   * **Encrypted**: brought in only as place holders - encrypted data expected to be inputted after by IT Pro.  Use **–ExportVars $True** or **–ExportAssets $True** to get all possible assets * **INT**, **BOOLEAN**, **String**, **DATE Time** variables brought in with Values, Description.  Use **–ExportVars $True** or **–ExportAssets $True** to get all possible assets * $**NULL** exported as referenced but currently exported as BOOL and not supported for import (shown below)   ***Note***    Encrypted data is not exported and will need to be entered manually (if not already present in the environment) upon import using this solution.  Additionally, encrypted variables are assumed to be of type string. ***Additional Note***    It is assumed that in a series of Runbooks, credentials and variables are defined according to the following format and ***not*** using string variables as names.  Get-AutomationPSCredential –name ‘SMA Credential Name’  and  Get-SMAVariable –name ‘SMA Variable Name’ |
| Credentials | Credentials will create placeholder credentials references in the exported XML.  Use **–ExportCreds $True** or **–ExportsAssets $True** to get all possible assets. |

### **Runbook Import**

Runbooks can be imported with the following options

|  |  |
| --- | --- |
| **COMPONENT** | **DESCRIPTION** |
| Published and Draft content | Published or Draft supported for import with **-RunbookState** “**Published**” or “**Draft**” |
| Runbook Tags | Runbook Tags (imported by default if present) |
| Runbook Description | Runbook Description (imported by default if present) |
| Runbook Logging | Runbook Logging options (imported by default if present) |
| Runbook Schedules | Runbook Schedules (Daily only) with –**ImportSchedules $True** or **–ImportAssets $True** to import all possible assets. |
| Leveraged SMA Variables | The following SMA Variables are supported for Import   * Encrypted (brought in only as place holders (encrypted data inputted after by IT Pro) using **–ImportVars $True** or **–ImportAssets $True** to import all possible assets * INT, BOOLEAN, String, DATE Time variables brought in with Values, Description (with **–ImportVars $True** or **–ImportAssets $True** to import all possible assets) * $NULL not supported due to limitation with cmdlets currently   ***Note***    Encrypted data is not exported and will need to be entered manually (if not already present in the environment) upon import using this solution.  Additionally, encrypted variables are assumed to be of type string and will be created as encrypted string variables on import. ***Additional Note***    It is assumed that in a series of Runbooks, credentials and variables are defined according to the following format and ***not*** using string variables as names (when exported).  Get-AutomationPSCredential –name ‘SMA Credential Name’  and  Get-SMAVariable –name ‘SMA Variable Name’ |
| Credentials | Credentials – will create **placeholder credentials** in SMA environment with **default credentials** (only if creds do not already exist) leveraging **–ImportCreds $True** or **–ImportAssets $True** to import all possible assets. |

### **Capabilities of Import / Export Solution (Features)**

|  |  |
| --- | --- |
| **FEATURE** | **DESCRIPTION** |
| PS1 included on Export | Include PS1 file as an export artifact (**-ExportPS1 $True**) |
| Quick Publish | Import Quick Publish (instead of bringing the Runbook in only in draft) by using the **–Publish $True** however it is recommended to use example **foreach** to publish post batch import to ensure parent / child Runbook relationship remains in tact |
| Overwrite Option | Import Overwrite option (if a Runbook already exists) using **–OverWrite $True**.  Default the Runbook will not get overwritten if it already exists in draft mode. |
| Batching for Runbook Projects | Runbooks can be imported / exported in batches leveraging the example Runbooks imported with the solution or via PowerShell with the examples in this post   * Exported by anything searchable related to the Runbook such as tag, Log setting, description, name, etc. * Runbooks can be imported by PS1 and XML (even if both exist within the same directory) |
| Native PowerShell only Import | Importing PS1 only brings in PowerShell |
| XML Import Supports Advanced Options | XML brings in all enhanced atomic options (shown above) |
| Export All Assets | Export all assets (that it currently can) by using the **–ExportAssets $True**  removing the requirement to specify var, cred, schedule options on the commandline. |

## Out of Scope for SMART for Runbook Import and Export

* Connections
* Secrets (that’s encrypted data)
* NULL Variables (not supported by the cmdlets to create from PowerShell currently)
* Updating TAGS of Published Runbooks (also not supported by cmdlets currently)

## Getting Things Setup

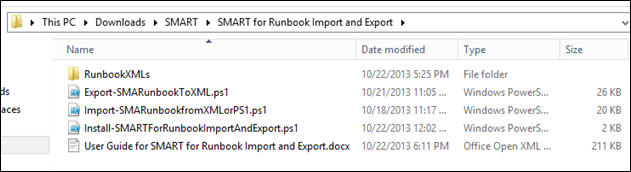
So how do I get this thing setup you ask?  Well, go out and download the content from the TechNet Gallery location, extract it and run it.

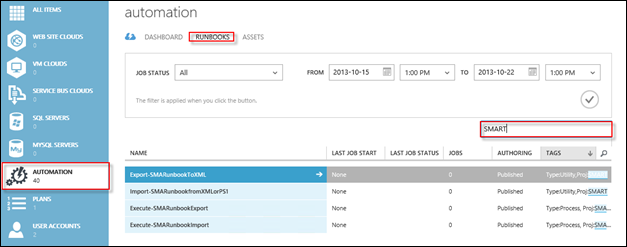
### Requirements

* This should be run from a system that has the SMA PowerShell provider installed.  This installation is available on the installation menu for Orchestrator.
* You’ll want to make sure you have a credential to execute against SMA or the existing credential you are logged in with should have sufficient rights to import / export contents from SMA.  Leveraging –cred option will provide the ability to enter a stored credential object from a PowerShell console.
* You should run this from an elevated “Administrator” PowerShell console or PowerShell ISE.

### Installation

To get the solution installed

* Extract the contents of the TechNet Gallery download to a system that has the PowerShell provider installed.   
  [](http://blogs.technet.com/cfs-file.ashx/__key/communityserver-blogs-components-weblogfiles/00-00-00-85-24-metablogapi/1185.image_5F00_0A4DAEE6.png)
* Execute the **Install-SMARTForRunbookImportAndExport.PS1** from an elevated PowerShell console or ISE to import the 4 Runbooks in the RunbookXMLs folder shown above into your SMA environment.
* The Result will be (**4**) Runbooks imported into your SMA environment that include Tags for easy searching
  + **Export-SMARunbookToXML**: This is essentially the Runbook version of the PowerShell version Export-SMARunbookToXML.PS1
  + **Import-SMARunbookfromXMLorPS1**: This is the Runbook version of Import-SMARunbookfromXMLorPS1.ps1
  + **Invoke-SMARunbookExport**: This Runbook provides a wrapper framework for executing an SMA Runbook Export
  + **Invoke-SMARunbookImport**: This Runbook provides a wrapper framework for executing an SMA Runbook Import

[](http://blogs.technet.com/cfs-file.ashx/__key/communityserver-blogs-components-weblogfiles/00-00-00-85-24-metablogapi/1018.image_5F00_4D72D6AE.png)

# Some Examples to Get You Started

Below are some examples to get you started.  Obviously, there is a lot of flexibility in this solution and it is really meant to provide a framework you can call into to get what you need.  You’ll see what I mean in the below examples. These are meant to be examples that you could run from a PowerShell script or directly inside the **Invoke-SMARunbookExport** and **Invoke-SMARunbookImport** Runbooks that you just installed.

## Export SMA Runbooks with SMART

#### Export a Single Runbook That has been Created within SMA (Simple Scenario)

Let’s start simple! The below example shows how you would leverage SMART to export a single Runbook from your SMA environment to include Tags, Description, Log Settings, and of course the PowerShell to an export XML. Nothing fancy like variables or credentials – just the basics.

|  |  |
| --- | --- |
| 001  002  003  004  005  006  007  008  009  010  011  012 | $WebServiceEndpoint = "https://localhost"  $ExportDirectory = "C:\temp\Runbook-Example"  $cred = Get-Credential   # Find the Runbook named Runbook-Example  $Runbook = Get-SmaRunbook -WebServiceEndpoint $WebServiceEndpoint `  -Credential $cred | Where-Object -Match -Property RunbookName "Runbook-Example"    .\Export-SMARunbooktoXML.ps1 -RunbookName $Runbook.RunbookName -ExportDirectory $ExportDirectory `      -WebServiceEndpoint $WebServiceEndpoint –Cred $cred –EnableScriptOutput $True `      -ExportPS1 $True –ExportVars $False –ExportCreds $False –ExportSchedules $False `      -ExportAssets $False |

And the results are shown below (XML result)

|  |  |
| --- | --- |
| 001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021 | <?xml version="1.0" encoding="UTF-8"?>  <Runbook>    <Name>Runbook-Example</Name>    <Tag>Proj:Runbook-Example</Tag>    <Configuration>      <Description>A simple Runbook Example</Description>      <LogDebug>False</LogDebug>      <LogVerbose>False</LogVerbose>      <LogProgress>False</LogProgress>    </Configuration>    <Published>      <Definition>workflow Runbook-Example  {      $Services = Get-Service      $Services  } </Definition>    </Published>    <Draft>      <Definition>Draft Not Unique</Definition>    </Draft>  </Runbook> |

#### Export all Runbooks that Match a Particular Tag in SMA (Advanced Scenario)

Ok, let’s get a bit fancier and go after a larger set of Runbooks.

Notice in the below example, this is actually running from a PowerShell script outside of SMA calling the PS1 file directly. A few things to note.

* + Mandatory variables are established that should be updated for your environment such as $ExportDirectory and $WebServiceEndpoint
  + This example is searching SMA by tags to find all Runbooks with a particular project name
  + –**ExportAssets $True** is specified ensuring you are exporting all assets with the Runbook on export
  + Other Boolean variables are specified with **$False** just to show other granular options
  + Assumption is you are executing this from a PS1 below from the working directory of SMART.

|  |  |
| --- | --- |
| 001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017 | $WebServiceEndpoint = "https://localhost"  $ExportDirectory = "C:\temp\SQLSSK"  $cred = Get-Credential   # Find all Runbooks matching a specific tag  $RunbooksToProcess = Get-SmaRunbook -WebServiceEndpoint $WebServiceEndpoint `  -Credential $cred | Where-Object -Match -Property Tags  "Proj:SQLSSK-"   foreach ($RB in $RunbooksToProcess)  {      .\Export-SMARunbooktoXML.ps1 -RunbookName $Rb.RunbookName -ExportDirectory $ExportDirectory `       -WebServiceEndpoint $WebServiceEndpoint -Cred $cred –EnableScriptOutput $True `       -ExportPS1 $True –ExportVars $False –ExportCreds $False –ExportSchedules $False `       -ExportAssets $True  } |

***Note***    The above example is also easily shown within the SMA Runbook ***Execute-SMARunbookExport***.  Key difference is updating the call to ***Export-SMARunbookfromXMLorPS1*** from a PS1 to a Runbook.

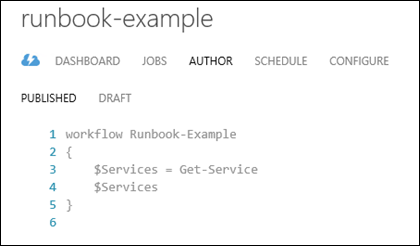
## Import SMA Runbooks with SMART

#### Import a Single Runbook from a Source Directory (Simple Scenario)

The below example takes a Runbook that has been exported with SMART into an XML and bring it into an SMA environment, leveraging the initial simple scenario I showed above for export.

|  |  |
| --- | --- |
| 001  002  003  004  005  006  007  008  009  010  011  012 | $WebServiceEndpoint = "https://localhost"   $ImportDirectory = "c:\temp\Runbook-Example"   # Specify extension! Very important.  $File = "Runbook-Example.XML"  $credential = Get-Credential    # $True and $False used to indicate available parameters. Not all required  .\Import-SMARunbookfromXMLorPS1.ps1 -ImportDirectory $ImportDirectory `  -FileName $File -overwrite:$True -RunbookState "Published" –ImportAssets $False `  -WebServiceEndpoint $WebServiceEndpoint -Port 9090 –ImportVars $False –ImportCreds $False `  -ImportSchedules $False -AuthenticationType Windows –cred $credential –Publish $True –EnableScriptOutput $True |

And the result of this import process

[](http://blogs.technet.com/cfs-file.ashx/__key/communityserver-blogs-components-weblogfiles/00-00-00-85-24-metablogapi/8233.image_5F00_2E874FCF.png)

### Import all Runbooks in a particular directory (Advanced Scenario)

Notice in the below example, this is actually running from a PowerShell script outside of SMA calling the PS1 file directly. A few things to note.

* Beginning of script establishes variables such as WebserviceEndpoint (your SMA Server) and the ImportDirectory (both are required)
* While loop is inserted to provide the option to import all Runbooks in a project and publish them successfully
* Each Boolean variable is shown with either a **$false** or **$true** ($false parameter/value pairs aren’t required but are shown to expose the available options).
* The below example will import all Runbooks or PS1 files in c:\temp\RunbookXMLs to Import
* The below example will import all defined configurations available (notice the **–ImportAssets $True** providing the ability to import all assets and not require the other asset parameters) in an XML or just the PS1 itself if an XML is not available.
* Notice that an optional **$EnableScriptOutput** is **not** set below.  This is on purpose since leveraging this in a batch operation and assigning to the Runbook array to publish would actually end up populating the array with script output and unpredictable results could result Smile.

|  |  |
| --- | --- |
| 001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030  031  032  033  034  035  036  037  038  039  040 | # IMPORT SECTION  #<#  # Bulk Import from Directory with XML Files  # Update $WebserviceEndpoint below if not running from the SMA server  # Update $ImportDirectory to reflect where the Runbook XMLs are located  $WebServiceEndpoint = "https://localhost"  $ImportDirectory = "c:\temp\RunbookXMLs to Import"  $Files = Get-ChildItem $ImportDirectory  $credential = Get-Credential   # Below while statement will go through twice to import/publish/edit/publish  # Looping twice is required to ensure parent and child Runbook relationships are  # properly established.  $i = 0  while($i -le 1)  {      $RunbookArray=@()      foreach ($File in $Files)      {          # $True and $False used to indicate available parameters. Not all required          $RunbookToPublish = .\Import-SMARunbookfromXMLorPS1.ps1 -ImportDirectory $ImportDirectory `          -FileName $File.Name -overwrite:$True -RunbookState "Published" –ImportAssets $True `          -WebServiceEndpoint $WebServiceEndpoint -Port 9090 –ImportVars $False –ImportCreds $False `          -ImportSchedules $False -AuthenticationType Windows –cred $credential           # Populate the Runbook listing for publishing          $RunbookArray+=$RunbookToPublish      }       # Publish all Runbooks      foreach($Runbook in $RunbookArray)      {          if($Runbook)          {              $PublishedRunbook= Publish-SMARunbook -Name $Runbook -WebServiceEndpoint $WebServiceEndpoint `              -Credential $credential -Port 9090 -AuthenticationType Windows          }      }      $i++  } |

***Note***    The above example is also easily shown within the SMA Runbook  ***Invoke-SMARunbookImport.*** Key difference is updating the call to ***Import-SMARunbookfromXMLorPS1*** from a PS1 to a Runbook as shown below

[image](http://blogs.technet.com/cfs-file.ashx/__key/communityserver-blogs-components-weblogfiles/00-00-00-85-24-metablogapi/6811.image_5F00_5D157270.png)

# Wrap Up

Ok, so you’ve gotten a taste of the solution and how it works.  There are more examples in the Synopsis included in the scripts and Runbooks. Just use get-help and leverage the –examples switch!

|  |  |
| --- | --- |
| 001  002 | get-help .\Export-SMARunbookToXML.ps1 -Examples  get-help .\Import-SMARunbookfromXMLorPS1.ps1 -Examples |

I’m definitely interested in the feedback on this solution and I hope you find the features and functionality useful for your Runbook management efforts.

# Examples

Even better, let me give you some examples you can chew on from some solutions developed within the team.  This will get you started taking a look at some valid scenarios you can not only use this tool with but also leverage.

* [Calling an Orchestrator Runbook from SMA – Part 1](http://blogs.technet.com/b/privatecloud/archive/2013/11/01/calling-an-orchestrator-runbook-from-sma-part-1.aspx):  Demonstrates how you can use your current existing Orchestrator Runbooks and integrate those with SMA.
* [SQL Server Self-Service Kit – Installation Instructions for the SMA Edition](http://blogs.technet.com/b/privatecloud/archive/2013/10/23/sql-server-self-service-kit-v2-0-installation-instructions.aspx): This post covers installation information for the SQL Server Self-Service Kit (SQLSSK) v2.0, a.ka. SMA Edition

**Until next time – Happy Automating!** Open-mouthed smile

And for more information, tips/tricks, and example solutions for SMA, be sure to watch for future blog posts in the [**Automation Track**](http://blogs.technet.com/b/privatecloud/archive/tags/automation+track/)!