

# PowerShell provider for BizTalk Server 2013

*Getting started guide*

version 1.4.0.1

---

Published: October 2014

Randal van Splunteren  
<http://biztalkmessages.vansplunteren.net>

Maxime Labelle  
<http://maximelabelle.wordpress.com>

# Contents

- Introduction ..... 4
- Installation and configuration ..... 5
  - Prerequisites** ..... 5
  - 32-bit or 64 bit mode**..... 5
  - Supported .Net Framework versions** ..... 5
  - Installation**..... 6
- Configuration ..... 7
  - Set PowerShell execution policy ..... 8
  - Load the BizTalk provider snap-in ..... 8
  - Default drive..... 8
  - Create a new PowerShell drive ..... 9
- Browsing BizTalk artifacts ..... 9
  - Browse the BizTalk artifact tree ..... 9
  - Combine with default cmdlets and pipelines ..... 11
- Using PowerShell built-in cmdlets ..... 13
  - Creating and removing items..... 13
  - Working with paths..... 14
  - Creating BizTalk artifacts..... 14
  - Removing BizTalk artifacts ..... 17
  - Getting and setting properties..... 18
- Using BizTalk Provider Cmdlets..... 19
  - Working with applications ..... 20
  - Add-ApplicationReference ..... 20
  - Remove-ApplicationReference ..... 20
  - Set-DefaultApplication ..... 20
  - Start-Application ..... 20
  - Stop-Application..... 21
  - Working with bindings ..... 22
  - Export-Bindings ..... 22
  - Import-Bindings..... 22

Working with Groups .....	23
Export-GroupSettings.....	23
Import-GroupSettings .....	23
Working with host instances.....	23
Restart-HostInstance.....	23
Start-HostInstance .....	23
Stop-HostInstance.....	23
Working with orchestrations .....	24
Enlist-Orchestration .....	24
Start-Orchestration .....	24
Stop-Orchestration.....	24
Unenlist-Orchestration.....	24
Working with policies.....	24
Export-Policy .....	24
Export-Vocabulary.....	25
Import-Policy.....	25
Deploy-Policy.....	25
Undeploy-Policy .....	25
Working with resources .....	25
Add-ApplicationResource.....	25
Get-ApplicationResourceSpec.....	26
Export-Application .....	26
Import-Application.....	26
Working with receive locations.....	28
Disable-ReceiveLocation .....	28
Enable-ReceiveLocation .....	28
Working with send ports.....	29
Enlist-SendPort.....	29
Start-SendPort.....	29
Stop-SendPort .....	29
Unenlist-SendPort .....	29
Working with send port groups .....	29
Enlist-SendPortGroup.....	29

Start-SendPortGroup.....	29
Stop-SendPortGroup.....	29
Unenlist-SendPortGroup.....	30
Appendix A: Document revisions.....	30

## Introduction

The PowerShell provider for BizTalk lets you manage Microsoft BizTalk Server from the command line. This guide helps you getting started with using the provider. It is by no means an exhaustive document as it only describes the basics. In a later phase we will provide a complete help file that describes all the cmdlets with their corresponding parameters.

In this release of the provider we do not cover all the options and features available to you when using the BizTalk Administration Console however we think it is still a valuable addition to the BizTalk toolset of both developers and administrators.

It is assumed that the reader of this document has at least basic knowledge of Microsoft Windows PowerShell as well as Microsoft BizTalk Server.

## Installation and configuration

In this first chapter we'll take the necessary steps to get the provider to run within a PowerShell session.

### Prerequisites

The provider requires the following software components:

- Microsoft Windows Operating system supporting PowerShell and BizTalk Server.
- Microsoft PowerShell 2.0 or 3.0.
- Microsoft BizTalk Server 2013 or 2013 R2.

*NB: older versions of the provider worked with older versions of BizTalk Server, including back to BizTalk Server 2006 R2. Those versions are now deprecated.*

The Microsoft BizTalk Server installation needs to be configured. The provider supports drive connections to both local and remote BizTalk installations. For a remote drive connection the machine running the provider has to be configured appropriately. This means DTC should be configured for remote access, etc.

### 32-bit or 64 bit mode

The provider is built on top of the BizTalk Management Automation library which depends on the **ExplorerOM** class that only runs in a 32 bit process. When you're using a 64 bit operating system you must be aware that you need to start the x86 (32 bit) version of the PowerShell interactive and scripting environment.



### Supported .Net Framework versions

The provider has been designed to operate against newer versions of Microsoft BizTalk Server, including version BizTalk Server 2013 and BizTalk Server 2013 R2, which both use assemblies built with .net framework 4.0.

By default, Windows PowerShell targets .net framework 2.0.

In order to be able to use the provider with BizTalk Server 2013 or BizTalk Server 2013 R2 assemblies, Windows PowerShell must be configured to explicitly support both versions of the .net framework. Therefore, the configuration file for Windows PowerShell must be updated like so:

- First, find the location of the Windows PowerShell configuration file:  
`%SystemRoot%\syswow64\WindowsPowerShell\v1.0\powershell.exe.config [x64]`  
`%SystemRoot%\system32\WindowsPowerShell\v1.0\powershell.exe.config [x86]`
- If this file does not exist, it must be created.

- In the configuration file, add the relevant section, highlighted in the following text:

```
<?xml version="1.0" encoding="utf-8"?>  
<configuration>  
  <startup>  
    <supportedRuntime version="v4.0" />  
  </startup> </configuration>
```

## Installation

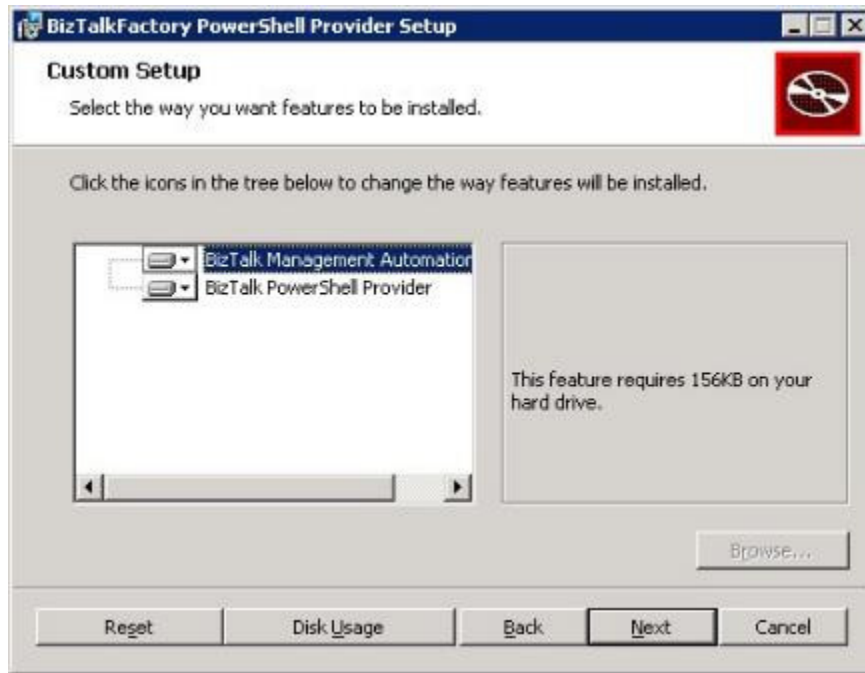
The first step of the installation is to download the provider installation package from Codeplex using the following address:

<http://psbiztalk.codeplex.com/releases/view/41461>

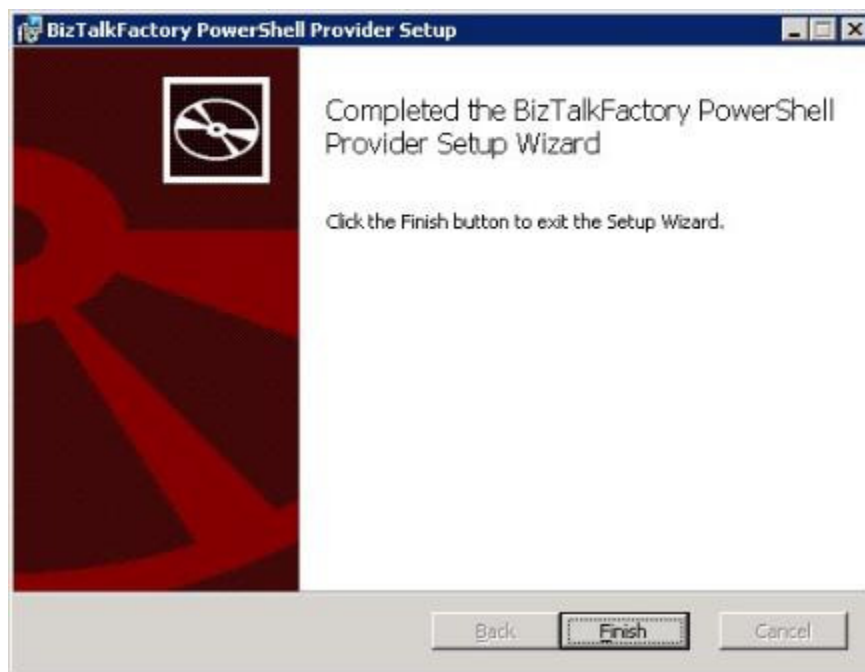
After downloading run the MSI. The following screen will appear:



The installation is straight forward. Follow the wizard to complete the installation of the provider.



When the provider has been successfully installed the following screen will appear:



Press the Finish button to exit the installation wizard.

### [Configuration](#)

The following paragraphs describe a couple of additional steps you will need to make before you can use the provider.

### Set PowerShell execution policy

The default execution policy in PowerShell is very secure. It does not allow for any scripts to run! In order to be able to work with the provider you should set the execution policy to *Unrestricted* or, better, *Remote Signed*<sup>1</sup>:

```
PS> Set-ExecutionPolicy -ExecutionPolicy RemoteSigned
```

### Load the BizTalk provider snap-in

Every PowerShell provider comes with a snap-in that loads the provider into the PowerShell session so you can work with it. To load the provider you need to execute the following statement:

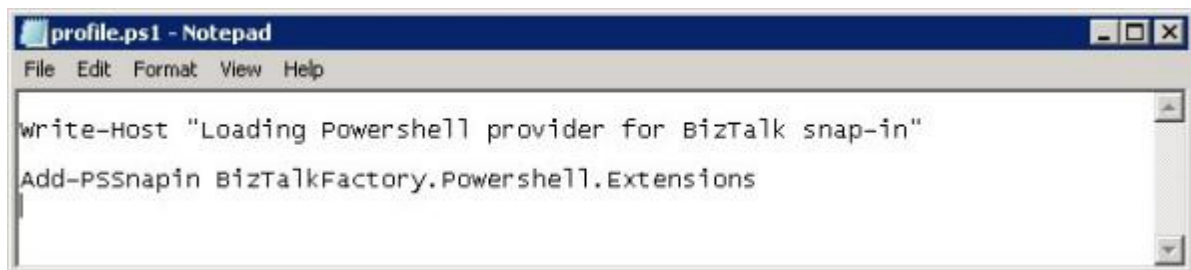
```
PS> Add-PSSnapin -Name BizTalkFactory.PowerShell.Extensions
```

Because you probably don't want to do this every time you can automate this step by adding it to your PowerShell profile. The PowerShell profile is a special kind of PowerShell script that runs when you start the PowerShell console or PowerShell integrated scripting environment (ISE). The profile scripts needs to be named 'profile.ps1' and needs to be in the following folder :

```
%HOMEPATH%\Documents\WindowsPowerShell
```

```
PS> New-Item (Split-Path $profile) -ItemType Directory
PS> notepad $profile
```

Below is an example of an appropriate `profile.ps1` file:



### Default drive

Upon loading, a default PowerShell drive will be created. The drive is called BizTalk and will point to a BizTalk management database called BizTalkMgmtDb on the local BizTalk installation. This is appropriate when you have a local single box installation of BizTalk.

When your SQL Server that hosts the BizTalk management database is remote or you have named your management database differently will fail and an exception will be shown in the PowerShell console.

To prevent this from happening, you can skip the initialization of the default drive by setting a specialized variable named `$InitializeDefaultBTSDrive` to `$false`.

---

<sup>1</sup> For more information on execution policy see:

<http://www.microsoft.com/technet/scriptcenter/topics/winpsch/manual/run.mspcx>



After you have skipped the default drive you will need to create a new PowerShell drive yourself. This is explained in the following paragraph.

### Create a new PowerShell drive

You can create a new PowerShell drive by providing the following information to the **New-PSDrive** cmdlet:

- Name of the drive
- Name of the BizTalk provider (value always “BizTalk”)
- Root of the drive (must match the name of the drive)
- Instance name of the SQL Server hosting the BizTalk management database
- Name of the BizTalk management database

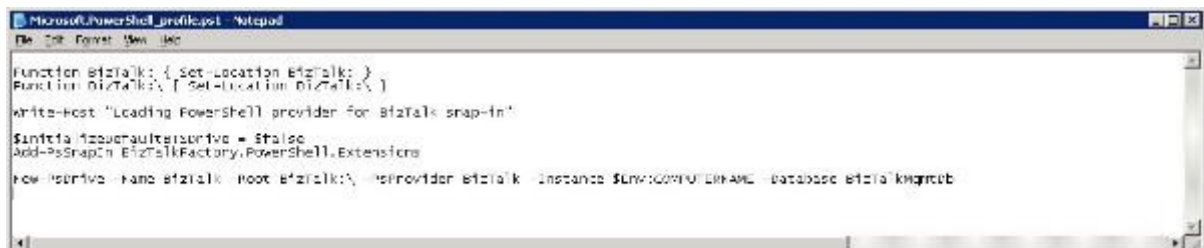
The example below will create a new drive BizTalk pointing to the BizTalk management MyBizTalkMgmtDb on MyServer.

```
PS> New-PSDrive -Name BizTalk -PSProvider BizTalk -Root BizTalk:\ `
    -Instance MyServer `
    -Database MyBizTalkMgmtDb
```

Use the *Remove-PSDrive* cmdlet to remove a drive:

```
PS> Remove-PSDrive -Name BizTalk
```

The `profile.ps1` script will look like this:



```
Microsoft.PowerShell_profile.ps1 - Notepad
File Edit Format View Help
Function BizTalk { Set-Location BizTalk }
Function BizTalk { [ Set-Location BizTalk ] }
Write-Host "Loading PowerShell provider for BizTalk snap-in"
$InitialPreference:Scripting = $False
Add-PSnapin BizTalkFactory.PowerShell.Extensions
New-PSDrive -Name BizTalk -Root BizTalk:\ -PSProvider BizTalk -Instance $ENV:COMPUTERNAME -Database BizTalkMgmtDb
```

**Known Limitation:** When using the *New-PSDrive* cmdlet, the name of the drive must exactly match the root portion of the drive. In other words, a drive named BTS would have a root BTS:\ whereas a drive named BizTalk would have a root BizTalk:\.

### Browsing BizTalk artifacts

In this chapter we are going to see how you navigate your way through the BizTalk artifacts. Samples in this chapter assume a local installation and the default BizTalk PowerShell drive called BizTalk.

#### Browse the BizTalk artifact tree

In the BizTalk PowerShell provider the artifacts are structured in the same way as in BizTalk Administration Console as much as possible. After opening PowerShell console you can switch to the default drive by typing:

```
PS> Set-Location -Path BizTalk:
```

```
BizTalk:\ >
```

You can make this command a little shorter by using the alias 'cd' and by using parameter order instead of named parameters:

```
PS> cd BizTalk: BizTalk:\  
PS>
```

As a convenience, you can write the preceding command in a PowerShell function that you make available to the session by storing it into the `$profile` file.

```
Function BizTalk: { Set-Location BizTalk: }
```

```
PS> BizTalk: BizTalk:\  
PS>
```

The prompt will now switch to the BizTalk drive and you can view its children by typing:

```
BizTalk:\ > Get-ChildItem
```

Or shorter:

```
BizTalk:\ > dir
```

After this command there are two items shown Applications and Platform Settings. You can switch to the Platform Settings by typing:

```
BizTalk:\ > Set-Location 'Platform Settings'  
BizTalk:\Platform Settings >
```

Note that the provider also supports tab completion. This means you can just type 'Plat' and press tab so that you don't have to type the complete statement.

Executing the `dir` statement again shows the children under the Platform Settings container. If you follow this by switching to the Hosts container and listing the children you get a list of all the hosts for the current BizTalk group.

```
BizTalk:\Platform Settings > Set-Location Hosts  
BizTalk:\Platform Settings\Hosts > Get-ChildItem
```

The screenshot below shows the console window after executing the above statements:

```

Administrator: Windows PowerShell (886)
PS BizTalk:\> Set-Location 'Platform Settings'
PS BizTalk:\Platform Settings> dir

    Path: BizTalkFactory.Powershell.Extensions\BizTalk::Biztalk:\Platform Settings

Name
----
Hosts
Host Instances

PS BizTalk:\Platform Settings> cd Hosts
PS BizTalk:\Platform Settings\Hosts> Get-ChildItem

    Path: BizTalkFactory.Powershell.Extensions\BizTalk::Biztalk:\Platform Settings\Hosts

Name                               Host Type                               Windows Group                           IsDefault
-----
BizTalkServerApplication           InProcess                               BizTalk Application Users               True
BizTalkServerIsolatedHost         Isolated                               BizTalk Isolated Host Users           False
myhost                             InProcess                               BizTalk Application Users              False
testhost                           InProcess                               BizTalk Application Users              False
testhost2                          InProcess                               BizTalk Application Users              False
trackinghost                       InProcess                               BizTalk Application Users              False

PS BizTalk:\Platform Settings\Hosts> _

```

As shown in the screenshot below you can do the same starting from Application level:

```

Administrator: Windows PowerShell (886)
PS BizTalk:\> Set-Location Applications
PS BizTalk:\Applications> Get-ChildItem

    Path: BizTalkFactory.Powershell.Extensions\BizTalk::Biztalk:\Applications

Name                               Status                               Description
-----
BizTalk.System                     NotApplicable                       System Application
BizTalk Application 1              NotApplicable                       BizTalk Application 1
MyBizTalkApp                       NotApplicable

PS BizTalk:\Applications> Set-Location MyBizTalkApp
PS BizTalk:\Applications\MyBizTalkApp> Get-ChildItem

    Path: BizTalkFactory.Powershell.Extensions\BizTalk::Biztalk:\Applications\MyBizTalkApp

Name
----
Orchestrations
Receive Ports
Receive Locations
Resources
Schemas
Pipelines
Send Ports
Send Port Groups

PS BizTalk:\Applications\MyBizTalkApp> _

```

### Combine with default cmdlets and pipelines

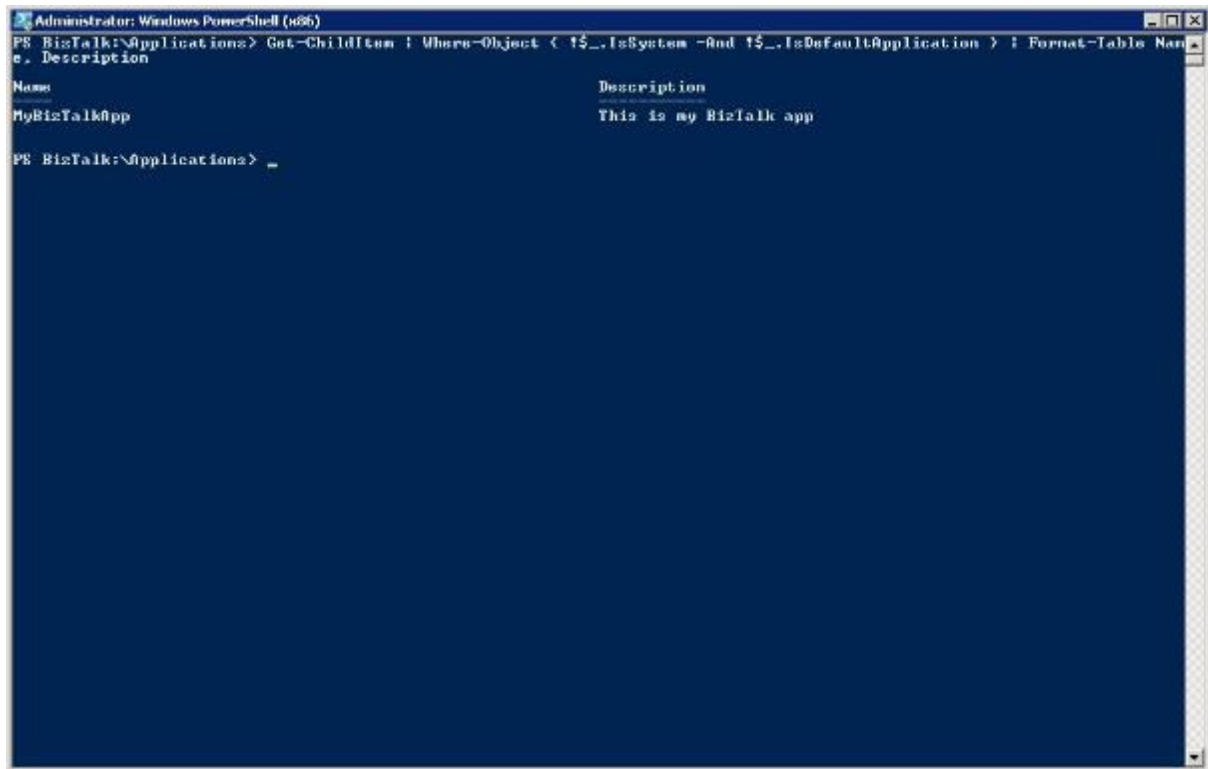
When combined with default PowerShell cmdlets and pipelines you can do more advanced stuff from the command prompt. For example let's say you want a list of application names and descriptions but you want to exclude the system and default application from this list:

```

BizTalk:\Applications > Get-ChildItem `
| Where-Object { !$_.IsSystem -And !$_.IsDefaultApplication } `

```

The output of this command is shown below. In this case there was only one application.

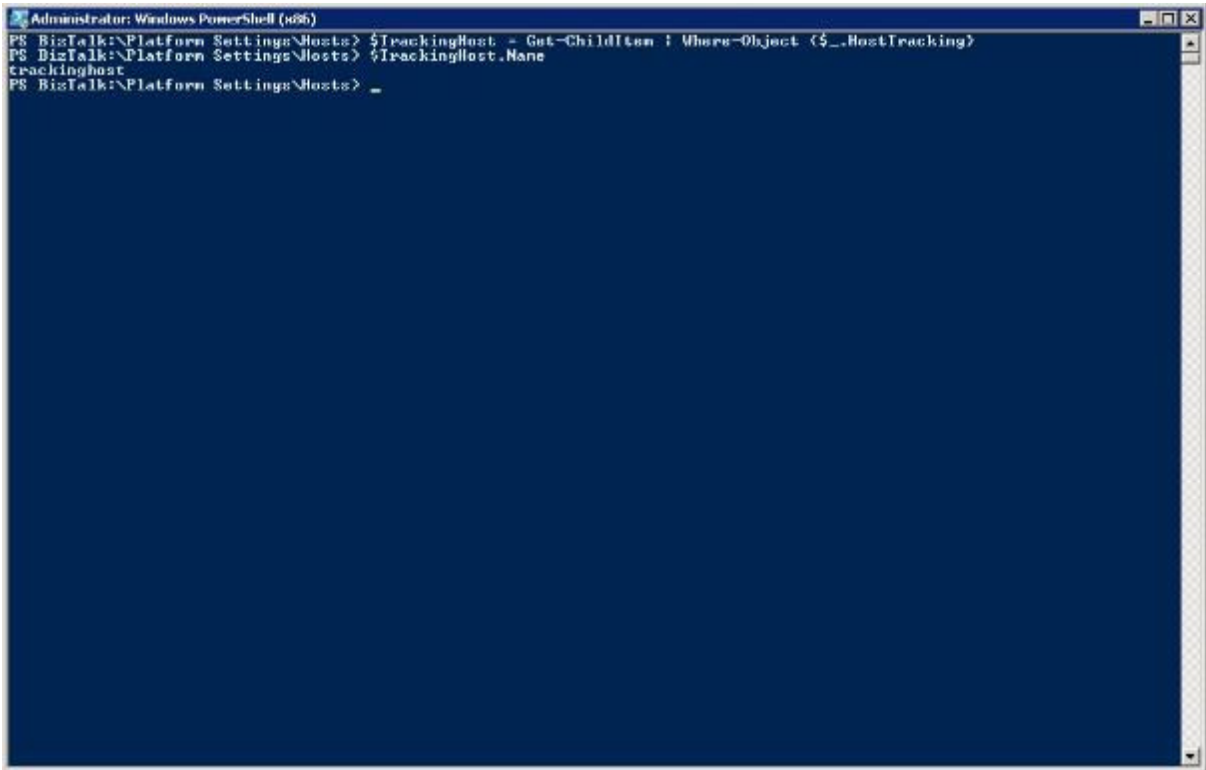


```
Administrator: Windows PowerShell (686)
PS BizTalk:\Applications> Get-ChildItem | Where-Object { !$_.IsSystem -And !$_.IsDefaultApplication } | Format-Table Name, Description
Name                                     Description
-----
MyBizTalkApp                             This is my BizTalk app
PS BizTalk:\Applications> _
```

In the following example we will execute a statement to find the host that is configured as the tracking host. When found we put this object in the variable \$TrackingHost. In the last line we use the variable to print the name of the host.

```
BizTalk:\Applications > $TrackingHost = Get-ChildItem | Where-Object { $_.HostTracking }
BizTalk:\Applications > $TrackingHost.Name
```

The above command in the console:



The final example retrieves the list of all currently running host instances and restarts them<sup>2</sup>.

```

BizTalk:\Applications > Set-Location '..\Platform Settings\Host Instances'
BizTalk:\Platform Settings\Host Instances > Get-ChildItem | `
Where-Object { $_.ServiceState -eq 'Running' } | `
Restart-HostInstance

```

### Using PowerShell built-in cmdlets

PowerShell comes with a set of default cmdlets that are implemented consistently across most providers. In the case of the provider for BizTalk you can create and remove artifacts.

You can also get or set a number of properties of those artifacts.

### Creating and removing items

This section shows how you can create and remove BizTalk artifacts using the provider. The following table shows which artifacts can be created and/or removed in this release:

BizTalk artifact	create	remove
Application	•	•
Host	•	•
Host instance	•	•
Orchestration		•
Pipeline		•
Policy		•
Receive handler	•	•
Receive location		•
Receive port		•

<sup>2</sup> This sample uses the Restart-HostInstance cmdlet which will be described later.

Resource	•	•
Schema		•
Send handler	•	•
Send port		•
Send port group		•

### Working with paths

Most cmdlets accept a Path parameter that specifies the name and location of the *item* to which the command applies. As is usual for interactive command prompts, paths can either be fully qualified or specified relative to the current working directory.

For instance, the following four commands are equivalent:

```
C:\Documents and Settings\JohnDoe\ > New-Item -Path BizTalk:\Applications\NewApplication
BizTalk:\ > New-Item -Path Applications\NewApplication
BizTalk:\Applications\ > New-Item -Path .\NewApplication
BizTalk:\applications\ > New-Item -Path NewApplication
```

The following examples in this guide must be understood in the context of an appropriate current directory.

### Creating BizTalk artifacts

To create a BizTalk artifact you will need to use the *New-Item* cmdlet. The command to create a new BizTalk application is:

```
PS> New-Item -Path MyNewApplication1 -Description 'This is a new Application'
```

The Description parameter is optional. Other optional parameters are:

- **Default**                Makes the new application the default application.
- **References**         Adds one or more references to other applications.

The next example will create a new application, set it as the default and add a reference to the previously created application.

```
PS> New-Item -Path MyNewApplication2 -Default -References MyNewApplication1
```

```

Administrator: Windows PowerShell (686)
PS BizTalk:\Applications> New-Item -Path MyNewApplication1 -Description 'This is a new Application'

Path: BizTalkFactory.Powershell.Extensions\BizTalk:\BizTalk:\Applications

Name                Status                Description
-----                -
MyNewApplication1   NotApplicable        This is a new Application

PS BizTalk:\Applications> New-Item -Path MyNewApplication2 -Default:$True -References MyNewApplication1

Path: BizTalkFactory.Powershell.Extensions\BizTalk:\BizTalk:\Applications

Name                Status                Description
-----                -
MyNewApplication2   NotApplicable

```

Using the *New-Item* cmdlet to add resources to an application is also possible. The following command adds a managed helper assembly to the current application:

```

PS> New-Item -Path MyResource -Source Z:\Assembly.dll `
           -ItemType System.BizTalk:Assembly `
           -GacOnAdd

```

The *ItemType* dynamic parameter is optional and indicates the type of resource that will be added. In most simple cases, the provider can infer the type from the *Path* parameters for the following types:

- System.BizTalk:Assembly                      Generic .net assembly.
- System.BizTalk: BizTalkAssembly            BizTalk assembly.
- System.BizTalk:Com                          In-process COM component.

All other resource types must be specified explicitly.

**Known Limitation:** the current version of the provider does not support resources typed System.Biztalk:Certificate.

The *SourceLocation* parameter is mandatory and specifies the source path for the resource being added. Its syntax depends upon the actual type of resource.

For instance, the following syntax must be used:

- System.BizTalk:Rule                          BRE ruleset Local Unique Identifier (Luid)
- System.BizTalk:WebDirectory                Uniform Resource Location (URI)

- *Other resource types*                      Local or UNC Path

Other optional parameters for this command are:

- Arguments                      Specifies arguments to pre- and post-processing scripts.
- DestinationLocation              Specifies the target location when used by .MSI installation.
- GacOnAdd                      Registers assembly to the global assembly cache upon add.
- GacOnImport                      Registers assembly to the GAC upon import.
- GacOnInstall                      Registers assembly to the GAC upon install.
- Regasm                      Registers assembly for depending COM clients.
- Regsvcs                      Registers assembly as a .Net Enterprise Services Component.
- RegSvr32                      Registers COM server into the Windows registry.
- TargetEnvironment              Associates a target environment to the bindings being added.

Two other artifacts you can create using the provider are hosts and host instances. The commands below create a host and corresponding host instance:

```
PS> New-Item -Path MyBizTalkHost -HostType InProcess -NtGroupName 'BizTalk Application
Users' -AuthTrusted
PS> New-Item -Path MyInstance -HostName MyBizTalkHost -RunningServer BTS2K9-dev
```

When applied to Hosts, the *New-Item* built-in cmdlet accepts the following additional dynamic parameters:

- HostType                      Creates either an *isolated* or *in-process* host.
- NtGroupName                      Specifies the group name for associated instances service accounts.
- AuthTrusted                      Specifies whether to create hosts trusted for authentication.

When applied to Host Instances, the *New-Item* built-in cmdlet accepts the following additional dynamic parameters:

- Credentials                      specifies logon credentials for the host instance service account.
- HostName                      specifies the name of the associated BizTalk host.
- RunningServer                      specifies the machine name where the host instance service will run.

**Known limitation:** The native `-Credential` parameter is not supported in any code cmdlets. Therefore, the provider for BizTalk uses a custom `-Credentials` parameter (note the extra 's' appended).

Note that in the preceding example we did not provide credentials for the host instance. Therefore, PowerShell shows a popup to enter the credentials<sup>3</sup>.

---

<sup>3</sup> There are also ways to provide the credentials from the command line and prevent the popup. For more information see: <http://blogs.msdn.com/powershell/archive/2008/06/20/getting-credentials-from->



```

Administrator: Windows PowerShell (686)
PS BizTalk:\Platform Settings\Hosts> New-Item -Path MyBizTalkHost -HostType InProcess -WlGroupName 'BizTalk Application Users' -AuthTrusted:$True

Path: BizTalkFactory.Powershell.Extensions\BizTalk::Biztalk:\Platform Settings\Hosts

Name                Host Type          Windows Group          IsDefault
-----                -
MyBizTalkHost       InProcess          BizTalk Application Users    False

PS BizTalk:\Platform Settings\Hosts> cd '..\Host Instances'
PS BizTalk:\Platform Settings\Host Instances> New-Item -Path MyInstance -HostName MyBizTalkHost -RunningServer B1S2M9-deu

cmdlet New-Item at command pipeline position 1
Supply values for the following parameters:
Credential:

Path: BizTalkFactory.Powershell.Extensions\BizTalk::Biztalk:\Platform Settings\Host Instances

Name                Host Name          Windows Group          Running Server          Host Type          Service State
-----                -
Microsoft BizTal... MyBizTalkHost       BizTalk Applicat...    B1S2K9-DEU             InProcess         Stopped

PS BizTalk:\Platform Settings\Host Instances> _

```

Finally, send handlers and receive handlers can also be created directly from the commandline. The following commands, run under an adapter subfolder, illustrate how:

```

BizTalk:\Platform Settings\Adapters\FILE\ > New-Item -Path MyHandler `
-HostName BizTalkSendHost `
-Direction Send

```

When applied to Receive or Send Handlers, the *New-Item* built-in cmdlet accepts the following additional dynamic parameters:

- **Direction** Creates either a *Receive* or *Send* handler.
- **HostName** Associates the handler to a BizTalk host.

### Removing BizTalk artifacts

To remove a BizTalk artifact we use the *Remove-Item* cmdlet accompanied with a path parameter. Because removing artifacts is straight forward and works the same for all artifacts we will only show an example where we remove an application.

```

PS> Remove-Item -Path MyNewApplication2 -Recurse

```

The *Recurse* parameter indicates we are aware that all children of this application – orchestrations, receive ports, pipelines, etc. – will also be removed.

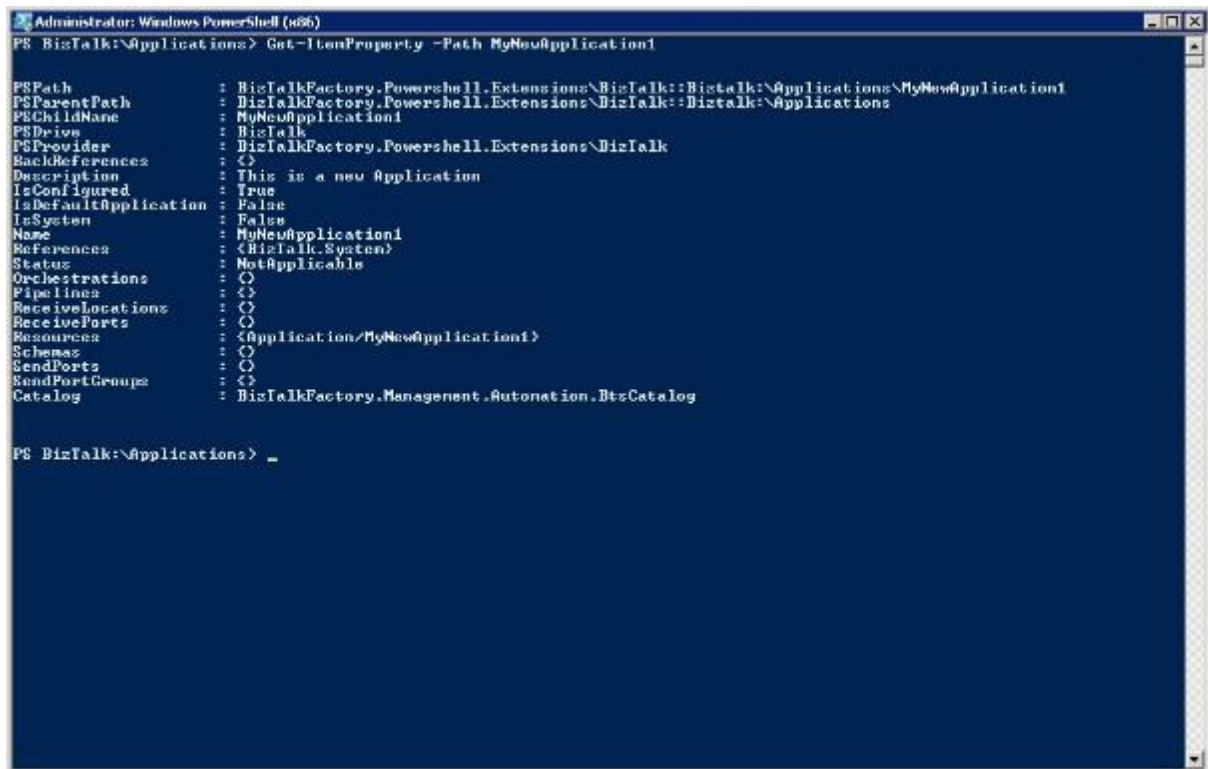
[thecommand-line.aspx](#) or <http://blogs.technet.com/robcost/archive/2008/05/01/powershell-tip-storing-andusing-password-credentials.aspx>

## Getting and setting properties

To get and set properties on a BizTalk artifact you can use the *Get-ItemProperty* and *Set-ItemProperty* cmdlets. The *Get-ItemProperty* cmdlet can be called with only a path parameter to fetch a set of all the properties of a certain artifact. To get all properties of an application:

```
PS> Get-ItemProperty -Path MyNewApplication1
```

The preceding command shows the following output:



```
Administrator: Windows PowerShell (x86)
PS BizTalk:\Applications> Get-ItemProperty -Path MyNewApplication1

PSPath                : BizTalkFactory.Powershell.Extensions\BizTalk::BizTalk:\Applications\MyNewApplication1
PSParentPath          : BizTalkFactory.Powershell.Extensions\BizTalk::BizTalk:\Applications
PSChildName           : MyNewApplication1
PSDrive               : BizTalk
PSProvider            : BizTalkFactory.Powershell.Extensions\BizTalk
BackReferences        : <>
Description           : This is a new Application
IsConfigured          : True
IsDefaultApplication  : False
IsSystem              : False
Name                  : MyNewApplication1
References            : <BizTalk.System>
Status                : NotApplicable
Orchestrations        : <>
Pipelines             : <>
ReceiveLocations     : <>
ReceivePorts         : <>
Resources             : <Application/MyNewApplication1>
Schemas              : <>
SendPorts             : <>
SendPortGroups       : <>
Catalog              : BizTalkFactory.Management.Automation.BtzCatalog

PS BizTalk:\Applications> _
```

You can get an individual property by providing its Name as a parameter:

```
PS> Get-ItemProperty -Path MyNewApplication -Name Description
```

```

Administrator: Windows PowerShell (886)
PS BizTalk:\Applications> Get-ItemProperty -Path MyNewApplication1 -Name Description

PSPath           : BizTalkFactory.Powershell.Extensions\BizTalk::BizTalk:\Applications\MyNewApplication1
PSParentPath     : BizTalkFactory.Powershell.Extensions\BizTalk::BizTalk:\Applications
PSChildName      : MyNewApplication1
PSDrive          : BizTalk
PSProvider       : BizTalkFactory.Powershell.Extensions\BizTalk
Description      : This is a new application

PS BizTalk:\Applications> _

```

Setting a property has almost the same syntax as getting a property except for the value parameter. For example to set the description of an application you use:

```
PS> Set-ItemProperty -Path MyNewApplication -Name Description -Value 'New description'
```

Another example to set the BizTalk group name:

```
PS> Set-ItemProperty -Path BizTalk:\ -Name Name -Value 'MyGroup'
```

## Using BizTalk Provider Cmdlets

Besides support for the built-in cmdlets as discussed in the previous chapter the PowerShell provider comes with an extensive set of BizTalk specific cmdlets. The following table provides an overview:

Working with...	Cmdlet	Description
Applications	Add-ApplicationReference	Add a reference to another application
	Remove-ApplicationReference	Remove a reference to an application
	Set-DefaultApplication	Set the default application
	Start-Application	Start an application
	Stop-Application	Stop an application
Bindings	Export-Bindings	Export bindings configuration to an XML file
	Import-Bindings	Import bindings configuration from a file
Orchestrations	Start-Orchestration	Start an orchestration
	Stop-Orchestration	Stop an orchestration
	Enlist-Orchestration	Enlist an orchestration
	Unenlist-Orchestration	Unenlist an orchestration

Resources	Add-ApplicationResource	Add a resource to an application
	Get-ApplicationResourceSpec	Retrieves the list of all resources
	Export-Application	Export an application as a .MSI package
Policies	Import-Application	Import an application from a .MSI package
	Deploy-Policy	Deploy a Business Rules Engine ruleset
	Export-Policy	Export a BRE ruleset to an XML file
	Import-Policy	Import a BRE ruleset from an XML file
	Undeploy-Policy	Undeploy a BRE ruleset
Receive Locations	Disable-ReceiveLocation	Disable a receive location
	Enable-ReceiveLocation	Enable a receive location
Send Ports	Start-SendPort	Start a send port
	Stop-SendPort	Stop a send port
	Enlist-SendPort	Enlist a send port
	Unenlist-SendPort	Unenlist a send port
Send Port Groups	Start-SendPortGroup	Start a send port group
	Stop-SendPortGroup	Stop a send port group
	Enlist-SendPortGroup	Enlist a send port group
	Unenlist-SendPortGroup	Unenlist a send port group
Host Instances	Start-HostInstance	Start a host instance
	Stop-HostInstance	Stop a host instance
	Restart-HostInstance	Restart a host instance

For each of the entries in the table you can find a sample below. These samples will only show a single way to use the cmdlets. For most cmdlets, there are multiple syntaxes. You can, for example, use PowerShell pipelines or provide additional parameters.

### Working with applications

Most of the custom cmdlets allow you to manage a BizTalk application.

#### Add-ApplicationReference

Add a reference to another application.

```
PS> Add-ApplicationReference -Path MyBizTalkApp -References MyNewApplication1, 'BizTalk Application 1'
```

#### Remove-ApplicationReference

Remove a reference to another application.

```
PS> Remove-ApplicationReference -Path MyBizTalkApp -References 'BizTalk Application 1'
```

#### Set-DefaultApplication

Set the default BizTalk application for this group.

```
PS> Set-DefaultApplication -Path MyBizTalkApp
```

#### Start-Application

Start an application with the provided options.

```
PS> Start-Application -Path MyBizTalkApp -StartOption StartAll
```

The following StartOption values are available:

- DeployAllPolicies
- EnableAllReceiveLocations
- StartAll
- StartAllOrchestrations
- StartAllSendPortGroups
- StartAllSendPorts
- StartReferencedApplications

### Stop-Application

Stop an application with the provided options.

```
PS> Stop-Application -Path MyBizTalkApp -StopOption StopAll
```

The following StopOption values are available:

- DisableAllReceiveLocations •  
StopAll
- StopReferencedApplications
- UndeployAllPolicies
- UnenlistAllOrchestrations
- UnenlistAllSendPortGroups
- UnenlistAllSendPorts

## Working with bindings

There are two cmdlets which help you to import and export binding files.

### Export-Bindings

Export a *bindings* file from either group, application or assembly level.

```
PS> Export-Bindings -Path MyBizTalkApp -Destination Z:\MyBindings.xml
```

The *Export-Bindings* cmdlet accepts the following additional parameters:

- GroupLevel Exports bindings for the whole BizTalk group.
- IncludeGlobalPartyBindings Includes configuration of all parties.

### Import-Bindings

Import a *bindings* file into the BizTalk group.

```
PS> Import-Bindings -Path MyBizTalkApp -Source Z:\MyBindings.xml
```

The *Import-Bindings* cmdlet accepts the following additional parameters:

- GroupLevel Imports bindings for the whole BizTalk group.

## Working with Groups

Cmdlets for exporting and importing Group Settings.

### Export-GroupSettings

Export settings about a BizTalk group.

```
PS> Export-GroupSettings -Path BizTalk:\ -Destination Z:\Temp\group_settings.xml
```

### Import-GroupSettings

Import settings into an existing BizTalk group.

```
PS> Import-GroupSettings -Path BizTalk:\ -Source Z:\Temp\group_settings.xml
```

## Working with host instances

Cmdlets for stopping, starting and restarting host instances.

### Restart-HostInstance

Restart a host instance.

```
PS> Restart-HostInstance -Path 'Microsoft BizTalk Server BizTalkServerApplication MyServer'
```

### Start-HostInstance

Start a host instance.

```
PS> Start-HostInstance -Path 'Microsoft BizTalk Server BizTalkServerApplication MyServer'
```

### Stop-HostInstance

Stop a host instance.

```
PS> Stop-HostInstance -Path 'Microsoft BizTalk Server BizTalkServerApplication MyServer'
```

## Working with orchestrations

You can manage orchestrations via the following cmdlets.

### Enlist-Orchestration

Enlist an orchestration.

```
PS> Enlist-Orchestration -Path MyProject.MyOrchestration
```

### Start-Orchestration

Start an orchestration.

```
PS> Start-Orchestration -Path MyProject.MyOrchestration
```

### Stop-Orchestration

Stop an orchestration.

```
PS> Stop-Orchestration -Path MyProject.MyOrchestration
```

### Unenlist-Orchestration

Unenlist an orchestration.

```
PS> Unenlist-Orchestration -Path MyProject.MyOrchestration
```

## Working with policies

You can deploy, undeploy, export or import Business Rules Engine rulesets and vocabularies with the following cmdlets.

There are two locations for working with policies and vocabularies:

- Under the “Policies” subfolder of a BizTalk Application folder.
- Under the “Policies” subfolder of the root “Rules Engine” folder.

For instance:

```
PS> Get-Item -Path BizTalk:\Applications\DefaultApplication1\Policies  
PS> Get-Item -Path 'BizTalk:\Rules Engine\Policies'
```

### Export-Policy

Export a *ruleset* file and, optionally, the dependent *vocabularies*.

```
PS> Export-Policy -Path MyPolicy -Version 1.0 -Destination Z:\MyPolicy.1.0.xml
```



The *Export-Policy* cmdlet accepts the following additional parameters:

- `AddReferencedVocabularies`      Includes referenced vocabularies.

### Export-Vocabulary

Export a *vocabulary* file and, optionally, the dependent *vocabularies*.

```
PS> Export-Vocabulary -Path MyVocabulary -Version 1.0 -Destination Z:\MyVocabulary.1.0.xml
```

The *Export-Vocabulary* cmdlet accepts the following additional parameters:

- `AddReferencedVocabularies`      Includes referenced vocabularies.

### Import-Policy

Import a *ruleset* file into the BizTalk group.

```
PS> Import-Policy -Path MyPolicy -Source Z:\MyPolicy.1.0.xml
```

### Deploy-Policy

Deploy a *ruleset* for use by BizTalk orchestrations.

```
PS> Deploy-Policy -Path MyPolicy -Version 1.0
```

### Undeploy-Policy

Undeploy a *ruleset* to the *published* state.

```
PS> Undeploy-Policy -Path MyPolicy -Version 1.0
```

## Working with resources

### Add-ApplicationResource

Add a resource stored in the file system to an application.

```
PS> Add-ApplicationResource -Path MyBizTalkApp -Type System.BizTalk: BizTalkAssembly  
-ResourcePath Z:\MySchemas.dll
```

**Warning:** The *Add-ApplicationResource* cmdlet is deprecated. Please, use the built-in **New-Item** cmdlet to handle the same situation.

### Get-ApplicationResourceSpec

Retrieve an XML representation of the resources in an application.

```
PS> Get-ApplicationResourceSpec -Path MyBizTalkApp
```

This cmdlet is useful when one wants to restrict the list of resources that must be included in a windows installer package. It can also be useful in order to retrieve the Local Unique Identifier for a specific resource.

The *Get-ApplicationResourceSpec* is usually followed by a call to the *Export-Application* below, in order to export a subset of all resources of an application, in an attempt to create incremental MSI packages :

```
PS> (Get-ApplicationResourceSpec -Path MyBizTalkApp).OuterXml | Out-File Z:\resspec.xml
```

**Warning:** make sure to use the *Out-File* built-in cmdlet to produce the UNICODE `resourcespec` file. Otherwise, an incorrect ANSI file would be created.

### Export-Application

Export an application MSI file to the file system.

```
PS> Export-Application -Path MyBizTalkApp -Package Z:\MyPackage.msi
```

The *Export-Application* accepts the following additional parameters:

- `IncludeGlobalPartyBindings` Includes configuration for all parties.
- `ResourceSpec` Specifies a custom list or resources to export.


In order to export only a subset of all available resources in an application, you can use the `ResourceSpec` parameter to specify an Xml representation of a list of resources to include in the resulting windows installer package like so:

```
PS> Export-Application -Path MyBizTalkApp `
    -Package Z:\MyPackage.msi `
    -ResourceSpec Z:\resspec.xml
```

### Import-Application

Import an application MSI from the file system into the BizTalk group.

```
PS> Import-Application -Path . -Package Z:\MyPackage.msi -Overwrite
```



The *Import-Application* accepts the following additional parameters:

- StagingEnvironment      Selects proper bindings with matching TargetEnvironment

## Working with receive locations

To enable or disable a receive location you can use:

### Disable-ReceiveLocation

Disable a receive location.

```
PS> Disable-ReceiveLocation -Path MyReceiveLocation
```

### Enable-ReceiveLocation

Enable a receive location.

```
PS> Enable-ReceiveLocation -Path MyReceiveLocation
```

## Working with send ports

For send ports there are four cmdlets provided

### Enlist-SendPort

Enlist a send port.

```
PS> Enlist-SendPort -Path MySendPort
```

### Start-SendPort

Start a send port.

```
PS> Start-SendPort -Path MySendPort
```

### Stop-SendPort

Stop a send port.

```
PS> Stop-SendPort -Path MySendPort
```

### Unenlist-SendPort

Unenlist a send port

```
PS> Unenlist-SendPort -Path MySendPort
```

## Working with send port groups

Finally the cmdlets to manage a send port group.

### Enlist-SendPortGroup

Enlist a send port group

```
PS> Enlist-SendPortGroup -Path MySendPortGroup
```

### Start-SendPortGroup

Start a send port group.

```
PS> Start-SendPortGroup -Path MySendPortGroup
```

### Stop-SendPortGroup

Stop a send port group.

```
PS> Stop-SendPortGroup -Path MySendPortGroup
```

## Unenlist-SendPortGroup

Unenlist a send port group.

```
PS> Unenlist-SendPortGroup -Path MySendPortGroup
```

## Appendix A: Document revisions

version	date	author	description
1.0	November 10th, 2009	Randal van Splunteren	Initial version
1.2	February 10th, 2011	Maxime Labelle	First official release
1.4.0.1	October 8th, 2014	Maxime Labelle	Updated for release 1.4.0.1.