PowerBroker Servers Windows Edition

Revision/Update Information: February 24, 2012
Software Version: PowerBroker Servers Windows Edition 1.1
Revision Number: 0

COPYRIGHT NOTICE
Copyright © 2012 BeyondTrust Software, Inc. All rights reserved. Use of this software and/or document, as and when applicable, is also subject to the terms and conditions of the license between the licencee and BeyondTrust Software, Inc. (“BeyondTrust”) or BeyondTrust’s authorized remarketer, if and when applicable.

TRADE SECRET NOTICE
This software and/or documentation, as and when applicable, and the information and know-how they contain constitute the proprietary, confidential and valuable trade secret information of BeyondTrust and/or of the respective manufacturer or author, and may not be disclosed to others without the prior written permission of BeyondTrust. This software and/or documentation, as and when applicable, have been provided pursuant to an agreement that contains prohibitions against and/or restrictions on copying, modification and use.

DISCLAIMER
BeyondTrust makes no representations or warranties with respect to the contents hereof. Other than, any limited warranties expressly provided pursuant to a license agreement, NO OTHER WARRANTY IS EXPRESSED AND NONE SHALL BE IMPLIED, INCLUDING WITHOUT LIMITATION THE WARRANTIES OF MERCHANTABILITY AND FITNESS FOR USE OR FOR A PARTICULAR PURPOSE.

LIMITED RIGHTS FARS NOTICE (If Applicable)
If provided pursuant to FARS, this software and/or documentation, as and when applicable, are submitted with limited rights. This software and/or documentation, as and when applicable, may be reproduced and used by the Government with the express limitation that it will not, without the permission of BeyondTrust, be used outside the Government for the following purposes: manufacture, duplication, distribution or disclosure. (FAR 52.227.14(g)(2)(Alternate II))

LIMITED RIGHTS DFARS NOTICE (If Applicable)
If provided pursuant to DFARS, use, duplication, or disclosure of this software and/or documentation by the Government is subject to limited rights and other restrictions, as set forth in the Rights in Technical Data – Noncommercial Items clause at DFARS 252.227-7013.

TRADEMARK NOTICES
PowerBroker, PowerPassword, and PowerKeeper are registered trademarks of BeyondTrust. PowerSeries, PowerADvantage, PowerBroker Password Safe, PowerBroker Directory Integrator, PowerBroker Management Console, PowerBroker Desktops, PowerBroker Virtualization, PowerBroker Express, PowerBroker Databases, PowerBroker Windows Servers, PowerBroker Windows Desktops, and PowerBroker Identity Services are trademarks of BeyondTrust. ssh® is a registered trademark of SSH Communications Security Corp in the United States and in certain other jurisdictions. The SSH logo, Tectia and tectia logo are trademarks of SSH Communications Security Corp and may be registered in certain jurisdictions. This application contains software powered by PKAIP®, the leading solution for enabling efficient and secure data storage and transmission. PKAIP® is provided by PKWARE, the inventor and continuing innovator of the ZIP file format. Used with permission.

FICTITIOUS USE OF NAMES
All names of persons mentioned in this document are used fictitiously. Any resemblance to actual persons, living or dead is entirely coincidental.
## Contents

**Introduction** ........................................................................................................ 9
  - Where to Go Next? ......................................................................................... 9
  - Documentation Set for PowerBroker Servers Windows Edition .................. 9
  - Getting Additional Help ............................................................................. 9

**Product Overview and Features** ...................................................................... 11
  - Product Overview ....................................................................................... 11
  - Product Features ......................................................................................... 11

**Getting Started with PowerBroker Servers** ............................................... 13
  - Concepts and Terms .................................................................................. 13
  - Architecture of PowerBroker Servers ....................................................... 15
  - How PowerBroker Servers Works ............................................................ 16

**Configuring PowerBroker Servers** .............................................................. 18
  - Working with a Run Host ........................................................................... 18
  - Working with the Proxy Host: Management Interface ............................ 18
  - Working with the Proxy Host: Policies ....................................................... 19
    - Defining Access Policies ....................................................................... 19
    - Defining Execute Policies .................................................................... 20
    - Querying Policies .................................................................................. 24
    - Removing Policies .................................................................................. 25
  - Viewing the Version of PowerBroker Servers ........................................ 26
  - Working with a Submit Host: Creating a Remote PowerShell Session .... 26

**Advanced Administrative Tasks** ................................................................. 28
  - Changing Elevation Credentials ............................................................... 28
  - Changing the PowerBroker Servers Port ................................................... 28

**Configuring Event Logging** ........................................................................ 31
  - Importing Event Logging Settings to the Domain Controller .................. 31
  - Configuring Event Logging Settings ......................................................... 33
  - Forwarding Events from the Authorization Agent .................................... 35

**Appendix A: Basic Commands** ................................................................. 36
  - Commands Required for Creating an Interactive Session ....................... 36
  - Commands Required for Auto-Completion .............................................. 37
  - Commands for Importing a Session .......................................................... 37
  - Troubleshooting Commands .................................................................... 37
  - ChildItem Commands .............................................................................. 38
    - Get-ChildItem ....................................................................................... 38
  - Content Commands .................................................................................. 38
<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add-Content</td>
<td>38</td>
</tr>
<tr>
<td>Clear-Content</td>
<td>39</td>
</tr>
<tr>
<td>Get-Content</td>
<td>39</td>
</tr>
<tr>
<td>Set-Content</td>
<td>39</td>
</tr>
<tr>
<td>Item Commands</td>
<td>40</td>
</tr>
<tr>
<td>Clear-Item</td>
<td>40</td>
</tr>
<tr>
<td>Copy-Item</td>
<td>40</td>
</tr>
<tr>
<td>Get-Item</td>
<td>41</td>
</tr>
<tr>
<td>Invoke-Item</td>
<td>41</td>
</tr>
<tr>
<td>Move-Item</td>
<td>41</td>
</tr>
<tr>
<td>New-Item</td>
<td>42</td>
</tr>
<tr>
<td>Remove-Item</td>
<td>42</td>
</tr>
<tr>
<td>Rename-Item</td>
<td>42</td>
</tr>
<tr>
<td>Set-Item</td>
<td>43</td>
</tr>
<tr>
<td>ItemProperty Commands</td>
<td>43</td>
</tr>
<tr>
<td>Clear-ItemProperty</td>
<td>43</td>
</tr>
<tr>
<td>Copy-ItemProperty</td>
<td>44</td>
</tr>
<tr>
<td>Get-ItemProperty</td>
<td>44</td>
</tr>
<tr>
<td>Move-ItemProperty</td>
<td>44</td>
</tr>
<tr>
<td>New-ItemProperty</td>
<td>45</td>
</tr>
<tr>
<td>Remove-ItemProperty</td>
<td>45</td>
</tr>
<tr>
<td>Rename-ItemProperty</td>
<td>45</td>
</tr>
<tr>
<td>Set-ItemProperty</td>
<td>46</td>
</tr>
<tr>
<td>Location Commands</td>
<td>46</td>
</tr>
<tr>
<td>Get-Location</td>
<td>46</td>
</tr>
<tr>
<td>Pop-Location</td>
<td>47</td>
</tr>
<tr>
<td>Push-Location</td>
<td>47</td>
</tr>
<tr>
<td>Set-Location</td>
<td>47</td>
</tr>
<tr>
<td>Path Commands</td>
<td>48</td>
</tr>
<tr>
<td>Join-Path</td>
<td>48</td>
</tr>
<tr>
<td>Convert-Path</td>
<td>48</td>
</tr>
<tr>
<td>Split-Path</td>
<td>48</td>
</tr>
<tr>
<td>Resolve-Path</td>
<td>49</td>
</tr>
<tr>
<td>Test-Path</td>
<td>49</td>
</tr>
<tr>
<td>PSDrive Commands</td>
<td>49</td>
</tr>
<tr>
<td>Get-PSDrive</td>
<td>50</td>
</tr>
<tr>
<td>New-PSDrive</td>
<td>50</td>
</tr>
<tr>
<td>Remove-PSDrive</td>
<td>50</td>
</tr>
<tr>
<td>PSPProvider Commands</td>
<td>51</td>
</tr>
<tr>
<td>Get-PSPProvider</td>
<td>51</td>
</tr>
<tr>
<td>Additional Commands</td>
<td>51</td>
</tr>
<tr>
<td>Get-Alias</td>
<td>51</td>
</tr>
<tr>
<td>Set-ExecutionPolicy</td>
<td>52</td>
</tr>
<tr>
<td>Get-Process</td>
<td>52</td>
</tr>
<tr>
<td>Get-Service</td>
<td>52</td>
</tr>
<tr>
<td>Get-Eventlog</td>
<td>53</td>
</tr>
</tbody>
</table>

BeyondTrust®          February 24, 2012

6
Function Examples

C:................................................................. 53
cd................................................................. 54
cd\................................................................. 54
help............................................................... 55
mkdir............................................................. 55

Application Examples

ipconfig....................................................... 56
ping.............................................................. 56
notepad......................................................... 56
calc............................................................ 57

Appendix B: PowerBroker Servers Events

Proxy Host Events............................................ 58
Event 10001, PBWS Proxy.................................. 58
Event 10002, PBWS Proxy.................................. 59
Event 10003, PBWS Proxy.................................. 59
Event 10004, PBWS Proxy.................................. 60
Event 10005, PBWS Proxy.................................. 60
Event 10006, PBWS Proxy.................................. 61
Event 10007, PBWS Proxy.................................. 62
Event 10008, PBWS Proxy.................................. 62
Event 10009, PBWS Proxy.................................. 63
Event 10010, PBWS Proxy.................................. 63
Event 10011, PBWS Proxy.................................. 64

Run Host Events............................................... 65
Event 20002, PBWS Authorization Manager............. 65
Event 20003, PBWS Authorization Manager............. 66
Event 20005, PBWS Authorization Manager............. 66
Event 20006, PBWS Authorization Manager............. 67

Appendix C: Troubleshooting.................................. 69

Troubleshooting the Proxy Host............................ 69
Troubleshooting the Run Host: PowerBroker Servers Authorization Agent... 70
Troubleshooting the Run Host: PBWS Service............. 71
Introduction

This guide shows system administrators and security administrators how to configure and use BeyondTrust PowerBroker Servers Windows Edition (PBWS). This guide provides an overview of how PowerBroker Servers works and instructions for PowerBroker Servers configuration and use.

Where to Go Next?


Documentation Set for PowerBroker Servers Windows Edition

The complete PowerBroker Servers Windows Edition documentation set includes the following:

- PowerBroker Servers Windows Edition Installation Guide
- PowerBroker Servers Windows Edition Administration Guide

Getting Additional Help

If you encounter problems that are not covered in the documentation, contact BeyondTrust technical support.

When contacting technical support, provide the following information:

- Your company name
- Telephone and email address where you can be contacted
- Description of the problem and the steps you have taken to resolve it

You can contact BeyondTrust technical support by email or through the BeyondTrust website. If you are located in the United States, you can also contact technical support by telephone. Support is staffed 24 hours per day, seven days per week.

**Telephone:** +1 800-234-9072

**Email:** pbws-support@beyondtrust.com

**Web:** To submit a support request online:

2. Click **Login** and log into the BeyondTrust website using the password provided to you by BeyondTrust.
3. After reading the Welcome message, scroll to the top of the BeyondTrust Partner Portal pane and click **Customer Support Center**.
4. Scroll down to the **Add/View Incidents** section and click the + icon.
5. In the **View Your Incidents** pane, click **Add Incident**, enter the details requested, and click **Submit Incident** to file your request for technical support.
Product Overview and Features

The following topics provide an overview of PowerBroker Servers Windows Edition and its features.

Product Overview

PowerBroker Servers Windows Edition delivers simplified privilege identity management (PIM) for Microsoft Windows Server based computers and applications that leverage Windows PowerShell—the emerging standard for contemporary Windows Server administration—to improve both security and auditing. This document assumes the reader is familiar with Microsoft PowerShell.

PowerBroker Servers enables Windows system administrators and security administrators to define policy for which administrators may run certain tasks (such as PowerShell cmdlets or functions) with elevated full administrative privilege.

The result is that responsibility for such tasks as adding administrator accounts and managing mailboxes can be safely assigned to the appropriate people without disclosing the full administrative password. The full power of full administrative access is thus protected from potential misuse or abuse outside their specified job responsibilities, such as stopping services, erasing disks, or doing more subtle damage.

Furthermore, PowerBroker Servers can provide augmented event logs that include all actions that were performed with elevated privilege and by whom they were performed. This audit trail, combined with the safe partitioning of full administrative privilege, provides an extremely secure means of controlling administrative access to computers running Windows Server and applications and meeting compliance driven auditing requirements.

Product Features

PowerBroker Servers provides the following features:

- Precision control over an administrators' access to PowerShell actions (such as cmdlets and functions)
- Provide augmented logs that record the details of who ran what PowerShell tasks and at what administrative level to better meet compliance auditing requirements
- Manages PowerShell actions for privileged administrators through a secure, centralized policy store
• Supports remote PowerShell-enabled products and applications (such as Microsoft Exchange)
• Minimally invasive end-administrator experience
• Provides scalable, robust enterprise deployment
• Simple policy authoring using PowerShell out-of-the-box policy cmdlets
**Getting Started with PowerBroker Servers**

The following topics introduce you to key terms and concepts for PowerBroker Servers Windows Edition, the architecture of PowerBroker Servers, and how PowerBroker Servers works.

### Concepts and Terms

To understand what PowerBroker Servers Windows Edition can do for your organization, you need to understand the following key terms.

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain Administrator</td>
<td>An individual who manages Microsoft domains or Active Directory.</td>
</tr>
<tr>
<td>Elevated Privilege Windows Administrator</td>
<td>An individual who manages Microsoft server and application deployments via a limited rights domain user account, where elevated privilege is granted using PowerBroker Servers.</td>
</tr>
<tr>
<td>Full Privilege Windows Administrator</td>
<td>An individual who manages Microsoft server or application deployments using a full administrative domain user account.</td>
</tr>
<tr>
<td>PowerShell Proxy Administrator</td>
<td>An individual who manages the PowerBroker Servers Proxy Host as a Full Privilege Windows Administrator.</td>
</tr>
<tr>
<td>domain controller (DC)</td>
<td>A computer that responds to security authentication requests (such as logging in and checking permissions) within the Windows domain. The domain controller is used by PowerBroker Servers to authenticate access to the Run Host or applications that reside on the Run Host.</td>
</tr>
<tr>
<td>Proxy Host</td>
<td>Also called PowerBroker Servers Proxy Host. The Proxy Host is the computer where the core PowerBroker Servers policy control engine resides. The Proxy Host intercepts remote PowerShell commands from any Submit Host that has had the appropriate web service redirected to the proxy.</td>
</tr>
<tr>
<td>Run Host</td>
<td>Also called PowerShell-enabled application or server. The Run Host is the computer where the application being controlled (such as Microsoft Exchange) resides. To follow a least privilege principle, access to the Run Host should be restricted and controlled by the PowerBroker Servers Proxy Host.</td>
</tr>
<tr>
<td>Submit Host</td>
<td>Also called administrator’s computer or administration client (application specific). The Submit Host is any computer that emits remote PowerShell tasks to a Run Host. PowerShell tasks may be emitted by either a Windows PowerShell command line environment or using a management application (such as Microsoft Exchange) that uses remote PowerShell “under the hood” of the applications administrative administrator interface.</td>
</tr>
<tr>
<td>PowerBroker Servers Management Interface</td>
<td>A PowerShell command line environment available on the PowerBroker Servers Proxy Host that is used to configure the PowerBroker Servers environment and policy.</td>
</tr>
<tr>
<td>Policy Enforcement Point (PEP)</td>
<td>A component of the PowerBroker Servers architecture that resides on the PowerBroker Servers Proxy Host. The PEP controls whether an administrator’s action from a Submit Host to a Run Host is allowed to run.</td>
</tr>
<tr>
<td>Policy Decision Point (PDP)</td>
<td>A component of the PowerBroker Servers architecture that resides on the PowerBroker Servers Proxy Host. The PDP evaluates whether an</td>
</tr>
<tr>
<td>Term</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>decision request (from a PEP to a PDP)</td>
<td>A communication, internal to the PowerBroker Servers architecture, in which the PEP asks the PDP whether an administrator's action from a Submit Host to a Run Host should be allowed to run.</td>
</tr>
<tr>
<td>Authorization Agent</td>
<td>A component of the PowerBroker Servers that resides on the Run Host. The Authorization Agent discovers whether a specific domain administrator is authorized to access a specific server or PowerShell action.</td>
</tr>
<tr>
<td>Access Policy</td>
<td>A high level PowerBroker Servers policy that defines whether a particular elevated privilege Windows administrator should have the ability to access a Run Host to execute tasks. (The tasks the administrator can issue must be defined explicitly in an Execute Policy.)</td>
</tr>
<tr>
<td>Execute Policy</td>
<td>A fine grained PowerBroker Servers policy that defines whether a particular elevated privilege Windows administrator should have the ability to run a specific action (such as a PowerShell cmdlet or function) on a remote host. (The administrator must have an explicit Access Policy that grants access to a Run Host prior to any Execute Policy). Typically, there is a one-to-one correspondence between an Execute Policy and the ability to run a specific task, and an administrative task will require one to many Execute Policies to perform the administrative task.</td>
</tr>
<tr>
<td>Policy Locator</td>
<td>A plugin extension to JBoss PDP (a component installed by PowerBroker Servers on the Proxy Host) used to find the eXtensible Access Control Markup Language (XACML) policy that the policy decision is made against. By default, JBoss PDP is configured using a file-based Policy Locator, and the entire XACML policy is stored in an XML file. PowerBroker Servers instead uses an HTTP-based Policy Locator that queries the specific HTTP service for the current policy.</td>
</tr>
</tbody>
</table>
Architecture of PowerBroker Servers

PowerBroker Servers Windows Edition uses a remote Windows PowerShell proxy architecture in which an administrator's PowerShell actions can be intercepted and controlled, allowing you to permit, deny, and elevate privilege. The functional components of the PowerBroker Servers architecture are described in “Concepts and Terms,” page 13.

Figure 1: High level architecture of PowerBroker Servers Windows Edition
How PowerBroker Servers Works

PowerBroker Servers Windows Edition is a solution that enables centralized Privilege Identity Management (PIM) functions for Windows Server operating systems and applications that use Windows PowerShell technology. The objective of the solution is to control the tasks that an administrator can perform on computers running the Windows Server operating system, as measured against pre-defined policy definitions.

Figure 2: Sample workflow for PowerBroker Servers Windows Edition

The following is the remote command execution flow, listing the typical interaction of the components of PowerBroker Servers. Each numbered step corresponds a number in the workflow diagram.

This execution flow begins when PowerShell commands from a Submit Host are configured to be re-directed to the PowerBroker Servers Proxy Host.

1. The Proxy Host intercepts remote PowerShell command request from the Submit Host to the Run Host.
   Before forwarding request to the Run Host, the Proxy Host checks whether the Submit User (access authentication stage) has access permission to perform the task measured against pre-defined policies (defined by the PowerShell Proxy Administrator using pre-defined policy authoring cmdlets).
2. The Proxy Host sends the policy request to the Policy Decision Point (PDP) to authorize the administrator (access authorization stage).

3. The PDP examines the rules defined in the stored policy and returns a decision.
   Before making each decision, the PDP sends a request to the PowerBroker Servers Proxy Service to get the current XACML policy (a policy definition understandable by JBoss PDP).

4. Upon request from PDP Policy Locator, the PowerBroker Servers Proxy Service transforms the policies defined by the pre-defined policy authoring cmdlets from domain-oriented representation into XACML policy.

5. The Proxy Host receives the response from the PDP and based on the answer performs one of the following actions:
   - Terminates execution of the remote commands, delivering an Access Denied error to the Submit Host if a policy blocks the action.
   - Elevates the authenticated Submit User to an Elevated Privilege Windows Administrator—a power user, typically a domain administrator—and forwards the remote commands entered on the Submit Host to the PowerShell runtime on the Run Host as if an Elevated Privilege Windows Administrator had entered the commands.

6. Native PowerShell runtime on the Run Host loads the pre-configured PowerBroker Servers Authorization Agent extension to PowerShell and sends each command to the PDP for evaluation of execute permission for this particular command.

7. The PDP examines the Master Policy Set (loaded at step 3) and responds with a decision.

8. Depending on the decision, the PowerBroker Servers Authorization Agent terminates execution or passes control to the PowerShell runtime to generate command output to the Submit Host.

9. The Proxy Host receives the response, logs the action, and forwards it to the Submit Host.

10. Permission is either granted or denied.
Configuring PowerBroker Servers

The following topics guide you through configuring a Run Host, a PowerBroker Servers Proxy Host, and a Submit Host.

Working with a Run Host

Prior to configuring a Submit Host or the PowerBroker Servers Proxy Host, you must install the PowerBroker Servers Authorization Agent on any Run Host that will be controlled by the PowerBroker Servers Proxy Host. For more information, see the PowerBroker Servers Windows Edition Installation Guide.

Working with the Proxy Host: Management Interface

All administrative configuration tasks for the PowerBroker Servers Proxy Host are performed using the PowerBroker Servers Management Interface, an administrative command line management shell that is installed during the Proxy Host installation.

To start the PowerBroker Servers Management Interface, in the Start menu in Windows, point to All Programs, BeyondTrust, PowerBroker Servers Windows Edition Management Shell. Alternatively, you can select the Launch PowerShell to configure PBWS option at the end of the Proxy Host installation.

After the PowerBroker Servers Proxy Host and Authorization Agent have been installed, it is important to configure the required Access Policies and Execute Policies on the Proxy Host that define which tasks a specific Elevated Privilege Windows Administrator can run on a specific Run Host (at full elevated privilege).

Tip: Getting command-line help for the PowerBroker Servers Management Interface

To get command-line help, including a description of the PowerBroker Servers Management Interface, cmdlet usage, and examples, execute the following:

help about_PBWS
Working with the Proxy Host: Policies

The concept behind PowerBroker Servers policy decision-making is that attribute values present in the incoming request are compared to the values that you specify in policy definitions. The execution of a requested action or access to a requested resource is permitted or denied based on the result of that comparison.

Accordingly, PowerBroker Servers Windows Edition differentiates between two types of policies:

- **Access Policy** – Defines which Run Hosts a particular administrator or group is or is not allowed to access.
- **Execute Policy** – Defines which applications, cmdlets, filters, and functions, are allowed or denied execution by a particular administrator on Run Hosts.

This differentiation allows you, for example, to temporarily deny access for an administrator without deleting his or her execution rights. When access is granted again, all the execution privileges are preserved.

To simplify the task of defining policy, PowerBroker Servers provides easy to use Windows PowerShell cmdlets (pronounced "command-lets") that define corresponding access or execution policies.

Defining Access Policies

To define an administrator’s access to Run Host, run the following Windows PowerShell cmdlet from the PowerBroker Servers Management Interface. This adds a new Access Policy record into the store.

**Syntax**

```powershell
Add-PowerBrokerPolicy
    -Computer <RunHostFQDN>
    -Principal <AdministratorName|GroupName>
    [-Effect <Permit|Deny>]
```

**Parameters**

- **-Computer <RunHostFQDN>**
  
  Required. Fully-qualified domain name (FQDN) of the Run Host that the administrator can access.

- **-Principal <AdministratorName|GroupName>**
  
  Required. An administrator or group for which you are going to allow or deny access to the Run Host.

  [-**Effect <Permit|Deny>**]
  
  Optional. Whether the access is permitted or denied. Default value is Permit.
Example 1
Add-PowerBrokerPolicy
    -Computer runhost.example.com
    -Principal EXAMPLE\Customers
Execution of this cmdlet creates a rule allowing members of the Customers group from the EXAMPLE domain to access the Run Host named runhost.example.com.

Example 2
Add-PowerBrokerPolicy
    -Computer runhost.example.com
    -Principal EXAMPLE\smith
    -Effect Deny
Execution of this command creates a rule restricting administrator smith from EXAMPLE domain from accessing the Run Host named runhost.example.com.

Tip: When rules conflict
Rules are combined by deny-overrides algorithm, so in case of two conflicting decisions, the result is Deny. For the preceding examples, if administrator smith is a member of Customers group, access to runhost.example.com is denied.

Defining Execute Policies
To define an administrator's execution rights on the Run Host, use the same Add-PowerBrokerPolicy Windows PowerShell cmdlet with different parameters. The following cmdlet adds a new Execute Policy record into the store.

Syntax
Add-PowerBrokerPolicy
    -Principal <AdministratorName|GroupName>
    -Command <CommandName>
    [-CommandType <Cmdlet|Application|Filter|Function>]
    [-CommandParameters @{ <parameter> = <value>; [ <parameter> = <value> ] }]
    [-Host <RunHostFQDN>]
    [-Effect <Permit|Deny>]

Parameters
    -Principal <AdministratorName|GroupName>
Required. An administrator or group that can access the Run Host.

-Command <CommandName>

Required. Name of the command that is denied or permitted for execution.

[-CommandType <Cmdlet|Application|Filter|Function>]

Optional. Command type. Default value is Cmdlet.

[-CommandParameters @{ <parameter>==<value>; [><parameter>==<value>]...}]}

Optional. To deny or permit execution of a command only when the command is run with specific parameter values, specify each parameter and an associated value. The parameters and values must be organized in a PowerShell hash table.

[-Host <RunHostFQDN>]

Optional. The fully-qualified domain name (FQDN) of a specific Run Host on which the administrator can or cannot execute the specified command. By default, the rule applies to all Run Hosts.

[-Effect <Permit|Deny>]

Optional. Whether the access is permitted or denied. Default value is Permit.

Example 1
Add-PowerBrokerPolicy
-Host runhost.example.com
-Principal EXAMPLE\Customers
-Command Clear-History
-Effect Deny

Execution of this cmdlet creates a rule preventing members of the Customers group in EXAMPLE domain from executing the Clear-History command on the Run Host named runhost.example.com.

Example 2
Add-PowerBrokerPolicy
-Principal EXAMPLE\smith
-Command Set-Location
-CommandParameters @{"Path"="windows"}
-Effect Deny

Execution of this cmdlet creates a rule restricting administrator smith of EXAMPLE domain from running the Set-Location command with the Path parameter set to windows.
Tip: How parameter values are evaluated

For the Add-PowerBrokerPolicy cmdlet, the values entered for parameters such as Principal, Command, and CommandParameters are treated as case-insensitive, but are compared as strings. If Example 2 is run, the administrator will not be able to execute Set-Location Windows, because windows and Windows are treated as equivalent regardless of case. However, the administrator can execute Set-Location C:\Windows, because C:\Windows and windows are not equivalent strings.

Using a Command Parameter Hash Table

You can deny or permit command execution in general and deny or permit the execution of specific parameters.

Example 3

This example permits the use of Get-Random regardless of the parameters used with that command:

Add-PowerBrokerPolicy -Principal EXAMPLE\smith -Command Get-Random -Effect Permit

Example 4

To permit or deny the use of specific parameters with command, define the parameters in a hash table, separating them with semicolons.

This rule denies execution of the Get-Random command when used with -Maximum 10 or -Minimum 2:

Add-PowerBrokerPolicy -Principal EXAMPLE\smith -Command Get-Random -CommandParameters @{Maximum="10"; Minimum="2"} -Effect Deny

The following examples show policies with different combinations of command and command parameters.

Example 5

If these rules are used, the Get-Random command is allowed for execution unless it is executed with -Maximum 10 or -Minimum 2 parameters. Execution is denied if any of the specified parameters are used in the execution attempt.

Add-PowerBrokerPolicy -Principal EXAMPLE\smith -Command Get-Random -Effect Permit
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
  -Command Get-Random
  -CommandParameters @{Maximum="10"; Minimum="2"}
  -Effect Deny

Example 6

If this rule is used, the Get-Random command is permitted for execution if at least one of the parameters in the hash table is specified with the command in the execution attempt, even if undefined parameters are included. If no parameters are defined, execution of the command will be denied:

Add-PowerBrokerPolicy -Principal EXAMPLE\smith
  -Command Get-Random
  -CommandParameters @{Maximum="10"}
  -Effect Permit

With this rule in effect, the following, the following outcomes result:

- Get-Random will be denied.
- Get-Random -Maximum 10 will be permitted.
- Get-Random -Maximum 10 -Minimum 3 will be permitted.
- Get-Random -Minimum 3 will be denied.

Example 7

If these rules are used, the Get-Random command is denied execution, regardless of whether it is run with or without parameters.

Add-PowerBrokerPolicy -Principal EXAMPLE\smith
  -Command Get-Random -Effect Deny

Add-PowerBrokerPolicy -Principal EXAMPLE\smith
  -Command Get-Random
  -CommandParameters @{Maximum="10"; Minimum="2"}
  -Effect Permit

Example 8

Add-PowerBrokerPolicy
  -Principal EXAMPLE\smith
  -Command test.exe
  -CommandType Application

This example creates an execution rule permitting administrator smith of EXAMPLE domain to run the test.exe application.
Tip: Allowing a Remote PowerShell session from a Submit Host to a Run Host

To create an interactive PowerShell session on a Submit Host, besides creating the corresponding access rule, you must permit execution of the following cmdlets for the target Run Host for the target Submit Host administrator:

- Out-Default
- Get-Command
- Set-StrictMode
- Measure-Object
- Select-Object
- Get-Help
- Test-Path
- ForEach-Object
- Get-Location
- Where-Object

You must also permit the execution of the following functions:

- Prompt
- TabExpansion

Everything that is not defined in the policies is denied for access and execution. Therefore, without a policy for these commands an administrator cannot start a remote PowerShell session. An example of defining these policies can be found in the PowerBroker Servers Sample Policy Library available from the BeyondTrust website.

Querying Policies

You can view the currently defined policies by using the Get-PowerBrokerPolicy Windows PowerShell cmdlet. You can filter the defined policies by using any of the pre-defined options that are available in Add-PowerBrokerPolicy.
Syntax
Get-PowerBrokerPolicy
[-Computer <RunHostFQDN>]
[-Principal <AdministratorName|GroupName>]
[-Command <CommandName>]
[-CommandType <Cmdlet|Application|Filter|Function>]
[-CommandParameters @{<parameter>=<value>; [<parameter>=<value>]...}]
[-Host <RunHostFQDN>]
[-Effect <Permit|Deny>]

Parameters

Example 1
To filter policies by a specific parameter, such as displaying all existing policies for the administrator named smith, you can use the following cmdlet:
Get-PowerBrokerPolicy -Principal EXAMPLE\smith

Example 2
You can combine filters by specifying several parameters. For example, the following cmdlet returns all of the policies that have a Permit effect for administrator smith.
Get-PowerBrokerPolicy -Principal EXAMPLE\smith
    -Effect Permit

Removing Policies
To remove a policy, execute the Remove-PowerBrokerPolicy Windows PowerShell cmdlet.

Syntax
Remove-PowerBrokerPolicy
    -Id <PolicyID>

Parameters
-Id <PolicyID>
    Required. ....

Example 1
To remove the policy that has an ID value of 2, run the following cmdlet.
Remove-PowerBrokerPolicy -Id 2
Example 2
If you are familiar with PowerShell pipes, you can perform `Remove-PowerBrokerPolicy` by instance using the PowerShell pipeline command. For information about how to use PowerShell pipes, see Microsoft TechNet.

The following example removes all records for administrator smith of EXAMPLE domain.

```
Get-PowerBrokerPolicy -Principal "EXAMPLE\smith" | Remove-PowerBrokerPolicy
```

Viewing the Version of PowerBroker Servers

You can obtain the version number of the instance of PowerBroker Servers Windows Edition installed on Proxy Hosts and Run Hosts by running the `Get-PowerBrokerVersion` Windows PowerShell cmdlet.

To get the PowerBroker Servers Proxy version number, execute the command from PowerBroker Servers Management Interface on the Proxy Host.

To get the PowerBroker Servers version number that is installed on the Run Host, execute the command in PowerBroker Servers remote session created to the corresponding Run Host via Proxy Host.

Syntax
`Get-PowerBrokerVersion [-verbose] [-debug]`

Parameters
- `-verbose`
  Optional.
- `-debug`
  Optional.

Working with a Submit Host: Creating a Remote PowerShell Session

Tip: Configure Proxy Settings on Submit Hosts

Before you can use a Submit Host to connect to a Run Host, you must configure proxy settings on the Submit Host. For details, see "Step 4: Configuring a Submit Host" in the *PowerBroker Servers Windows Edition Installation Guide*.
Administrators can execute commands on a Run Host by using an invoke-command pattern or by creating an interactive session from a Submit Host.

**Example 1**
The following script returns the Run Host name when executed from a Submit Host by an administrator with appropriate access and execution policies.

```powershell
Invoke-Command -ComputerName runhost.example.com
  -Authentication Kerberos
  -ScriptBlock
  { $Computer = Get-WmiObject -Class Win32_ComputerSystem;
    "Computer Name is: {0}" -f $Computer.Name}
```

**Example 2**
The following example creates a remote PowerShell session, enters it, and creates an interactive session on the Run Host.

```powershell
$session = New-PSSession -ComputerName runhost.example.com
  -Authentication Kerberos
Enter-PSSession $session
```

The difference between these examples is that in the first example having executed the command on the Run Host the administrator returns to the Submit Host, whereas in the second example the administrator enters the Run Host and all the consecutive commands are executed on the Run Host until the administrator exits the remote session.
Advanced Administrative Tasks

The following are advanced administrative tasks that you can perform using PowerBroker Servers Windows Edition.

Changing Elevation Credentials

You can change the credentials that the Proxy Host uses to provide elevation for all administrators that use Submit Hosts to connect to the Run Host.

To change the credentials used for elevation, on the Proxy Host:

1. From a Windows command prompt, navigate to the following folder:
   
   C:\Program Files\BeyondTrust\PowerBroker for Windows Server\Support Tools

2. In this folder, run the following command, substituting the account details to be used to provide elevation in place of the italicized text:
   
   aspnet_setreg.exe
   
   -k:SOFTWARE\BeyondTrust\PBWS\identity
   
   -u:"DomainName\AdministratorName"
   
   -p:"password"

3. Run the regedit command to open the Registry Editor.
4. In the Registry Editor, navigate to the following registry key:
   
   HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\BeyondTrust\PBWS\identity\ASPNET_SETREG

5. In the ASPNET_SETREG key, find NETWORK SERVICE and add Read permission for this account.
6. Close the Registry Editor.
7. At a Windows command prompt, run the following command so that the changes take effect:
   
   iisreset

Changing the PowerBroker Servers Port

By default, the Proxy Host and Submit Hosts use port 8989 to communicate. However, you can change which port they use.

To change the port on the Proxy Host that Submit Hosts use to communicate with the Proxy Host:

1. Change the port for the Proxy Host in IIS Manager:
   a. Open IIS Manager.
   b. Navigate to the Sites node and expand it to display the sites.
c. Right click the PBWS site and select **Edit Bindings**.
d. In the Site Bindings dialog, select the site binding used by PowerBroker Servers. By default, PowerBroker Servers uses port 8989. Click **Edit**.
e. In the Edit Site Binding dialog, change the **Port** value to a port of your choice and click **OK**.

![Site Bindings Dialog](image)

f. In the Site Bindings dialog, click **Close**.
g. From a Windows command prompt, run the `iisreset` command to restart IIS.

2. Change port for the Policy Locator:
   a. On the Proxy Host, navigate to `C:\JBoss\server\default\conf`.
   b. Right-click the `policyConfig` file and select **Properties**.
   c. In the Properties dialog, clear the **Read-Only** option and then click **OK**.
   d. Open the `policyConfig` file in a text editor such as Notepad and change port value to the new port that you specified in IIS Manager. The following is an excerpt from a typical configuration file in which the port is set to port 8989.

```xml
<ns:jbosspdp xmlns:ns="urn:jboss:xacml:2.0">
  <ns:Locators>
    <ns:Locator>
      Name="com.beyondtrust.powerbroker.locators.policy.SqlPolicyLocator">
        <ns:Option>
          Name="CacheLifeTime">30</ns:Option>
```
<ns:Option
  Name="WcfServiceURL">http://localhost:8989/admin/PolicyLocator.svc</ns:Option>
</ns:Locator>

<ns:Locator
  Name="com.beyondtrust.powerbroker.locators.attrib.ADMembershipAttributeLocator">
  <ns:Option
    Name="CacheLifeTime">3600</ns:Option>
</ns:Locator>

</ns:Locators>
</ns:jbosspdp>

e. Save and close the configuration file.
f. Restart the JBoss service, **JBoss Application Server 6.1**.

3. Add a Firewall exception for the new port.
4. On each Submit Host, change the proxy settings to correspond to the new port by running one of the following commands from an account that has local administrator privileges.
   • If the computer is running Windows 7, Windows Vista, Windows Server 2008 R2, or Windows Server 2008, run the following command at a Windows command. Substitute the IP address of the of the Proxy Host for `ProxyHostIP`, and the new port for `NewPort`.
     ```
     netsh winhttp set proxy proxy-server = ProxyHostIP:NewPort
     ```
   • If the computer is running Windows XP, Windows Server 2003 R2, or Windows Server 2003, run the following command at a Windows command prompt. Substitute the IP address of the of the Proxy Host for `ProxyHostIP`, and the new port for `NewPort`.
     ```
     proxycfg -p ProxyHostIP:NewPort
     ```
Configuring Event Logging

This section describes how to enable and configure event logging in PowerBroker Servers and the different types of events that can be logged. PowerBroker Servers events are logged in the Application Log of the appropriate host (Proxy Host or Run Host). You can choose which event types are logged.

The following is an example of a logged PowerBroker Servers event:

> Importing Event Logging Settings to the Domain Controller

To enable event logging, import the appropriate Group Policy Administrative Template for PowerBroker Servers into the Group Policy Management Editor. Group Policy is distributed to each computer on the domain and is automatically updated at regular intervals. To force a Group Policy update, run the `gpupdate /force` command from a Windows command prompt.

To enable event logging, you must import the PowerBroker Servers Group Policy Administrative Template:

1. If the functional level of the domain is Windows Server 2008:
   a. Copy the following ADMX file from the Proxy Host and save it to the domain controller (DC):
Proxy Host file location: C:\Program Files\BeyondTrust\PowerBroker for Windows Server\Support Tools\GPO\PBWS.admx

DC file location: C:\Windows\PolicyDefinitions\PBWS.admx

b. Copy the following ADML file from the Proxy Host and save it to the DC:
Proxy Host file location: C:\Program Files\BeyondTrust\PowerBroker for Windows Server\Support Tools\GPO\PBWS.adml
DC file location: C:\Windows\PolicyDefinitions\en-US\PBWS.adml

2. If the functional level of the domain is Windows Server 2003, copy the following ADM file from the Proxy Host and save it to any folder on the domain controller (DC):
Proxy Host file location: C:\Program Files\BeyondTrust\PowerBroker for Windows Server\Support Tools\GPO\PBWS.adm

a. On the DC, click Start, point to Administrative Tools, and click Group Policy Management to open the Group Policy Management Console (GPMC).
b. In the console tree of the GPMC, expand Group Policy Objects, and either double-click an existing GPO to edit it or else create and edit a new GPO.
c. In the Group Policy Management Editor (formerly the Group Policy Object Editor), right-click Administrative Templates and click Add/Remove Templates.
d. In the Add/Remove Templates dialog, click Add, select the ADM file that you copied to the DC, and click Open.

The new policy settings are displayed in the Administrative Templates node in the GPMC.

- If you have imported ADMX and ADML files, the policy settings appear under Computer Configuration, Policies, Administrative Templates, BeyondTrust, PBWS.
- If you have imported an ADM file, the policy settings appear under Computer Configuration, Policies, Administrative Templates, Classic Templates, BeyondTrust, PBWS.
Configuring Event Logging Settings

For most types of events, you can choose whether to enable logging for that type of event, and you can configure other options (such as whether to log successes, failures, or both).

To configure event logging:

1. On a computer from which you can manage Group Policy, open the Group Policy Management Console (GPMC), located in Administrative Tools.
2. In the console tree of the GPMC, expand Group Policy Objects, and double-click the Default Domain Policy GPO to edit it.
3. In the Group Policy Management Editor (formerly the Group Policy Object Editor), expand Computer Configurations, Policies, Administrative Templates, Classic Administrative Templates (if you imported an ADM file), BeyondTrust, PBWS.
4. Expand the node that corresponds to the category of event types that you want to configure—Proxy or Run Host—and then expand the Logging node.
a. Double-click a policy setting in the details pane to edit it.

b. In the Properties dialog for the policy setting, select either **Enabled** to turn on logging for the event type, or **Disabled** to turn off logging for the event type.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Applies to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Succeeded</td>
<td>Log success events (authentication, authorization, or Security Token Service (STS) elevation)</td>
<td>Authentication events, authorization events, STS events</td>
</tr>
<tr>
<td>Failed</td>
<td>Log failure events (authentication, authorization, STS elevation)</td>
<td>Authentication events, authorization events, STS events</td>
</tr>
<tr>
<td>Commands Audit</td>
<td>Log command executed for different command types</td>
<td>Authorization events</td>
</tr>
<tr>
<td>Denied by system</td>
<td>Log denial by PowerShell Authorization Manager</td>
<td>Authorization events</td>
</tr>
<tr>
<td>Denied by PowerBroker Servers</td>
<td>Log denial by PowerBroker Servers Authorization Agent</td>
<td>Authorization events</td>
</tr>
<tr>
<td>Created</td>
<td>Log successful PowerShell session creation</td>
<td>Proxy session events</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
<td>Applies to</td>
</tr>
<tr>
<td>--------</td>
<td>--------------------------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>Failed</td>
<td>Log failed PowerShell session creation event</td>
<td>Proxy session events</td>
</tr>
<tr>
<td>Ended</td>
<td>Log terminated PowerShell session event</td>
<td>Proxy session events</td>
</tr>
<tr>
<td>Errors</td>
<td>Log Web Services-Management (WS-Management) error event</td>
<td>WS-Management error events</td>
</tr>
</tbody>
</table>

c. If you have enabled the policy setting, you can select options to configure what is logged. The options available vary with the policy setting.

d. You can click **Next Setting** or **Previous Setting** to configure other policy settings in this collection.

e. When you are finished, click **OK** to save what you have configured.

**Forwarding Events from the Authorization Agent**

It is recommended that you use Event Forwarding to redirect events from the Proxy Host and Run Host to a single event log storage location. For more information, see the following topic on MSDN:

Appendix A: Basic Commands

The following is a list of basic cmdlets, functions, and applications. For each of these commands, an example is provided of either a PowerBroker Servers policy to be added when the command is executed in an interactive session, an example of a PowerBroker Servers policy to be added when the command is executed using invoke-command, or both.

Commands Required for Creating an Interactive Session

The following Windows PowerShell cmdlets and function must be allowed for the user to be able to create an interactive session.

- Out-Default
- Get-Command
- Set-StrictMode
- Measure-Object
- Select-Object
- Get-Help
- Test-Path
- ForEach-Object
- Get-Location
- Where-Object
- prompt

Policy Example (Enter-PSSession)

Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Out-Default
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Get-Command
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Set-StrictMode
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Measure-Object
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Select-Object
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Get-Help
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Test-Path
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command ForEach-Object
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Get-Location
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Where-Object
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command prompt -CommandType Function

Commands Required for Auto-Completion

The following cmdlets and function must be allowed for the user to be able to auto-complete commands, parameters, and variables.

- Split-Path
- Resolve-Path
- TabExpansion
- Sort-Object
- Get-ChildItem

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Split-Path
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Resolve-Path
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command TabExpansion -CommandType Function
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Sort-Object
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Get-ChildItem

Commands for Importing a Session

It is recommended that the following Windows PowerShell cmdlet be allowed for use when importing a session.

- Get-FormatData

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Get-FormatData

Troubleshooting Commands

The following Windows PowerShell cmdlet is a general command that can be helpful to have allowed when using invoke-command. It can help ensure that incorrect use of a command (such as incorrect syntax) returns the correct error.

- Set-StrictMode
Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Set-StrictMode

ChildItem Commands
The following are examples of policies that you can create to support the execution of ChildItem cmdlets in an interactive session or by using invoke-command.

Get-ChildItem

Aliases
dir, gci, ls

Command Type
Cmdlet

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Get-ChildItem

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Get-WmiObject
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Get-ChildItem

Content Commands
The following are examples of policies that you can create to support the execution of Content cmdlets in an interactive session or by using invoke-command.

Add-Content

Aliases
ac

Command Type
Cmdlet

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Add-Content
Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Add-Content

Clear-Content

Aliases
clc

Command Type
Cmdlet

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Clear-Content

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Clear-Content

Get-Content

Aliases
gc, cat, type

Command Type
Cmdlet

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Get-Content

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Get-Content
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Get-WmiObject
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Get-Content

Set-Content

Aliases
ac

Command Type
Cmdlet
Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Set-Content

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Set-Content

Item Commands
The following are examples of policies that you can create to support the
execution of Item cmdlets in an interactive session or by using invoke-
command.

Clear-Item

Aliases
cli

Command Type
Cmdlet

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Clear-Item

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Clear-Item

Copy-Item

Aliases
copy, cp, cpi

Command Type
Cmdlet

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Copy-Item

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Copy-Item
Get-Item

Aliases
gi

Command Type
Cmdlet

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Get-Item

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Get-WmiObject
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Get-Item

Invoke-Item

Aliases
ii

Command Type
Cmdlet

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Invoke-Item

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Invoke-Item

Move-Item

Aliases
move, mv, mi

Command Type
Cmdlet

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Move-Item
Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Move-Item

New-Item

Aliases
ni

Command Type
Cmdlet

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command New-Item

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Get-WmiObject
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command New-Item

Remove-Item

Aliases
del, rd, erase, ri, rm, rmdir

Command Type
Cmdlet

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Remove-Item

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Remove-Item

Rename-Item

Aliases
ren, rni

Command Type
Cmdlet
Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Rename-Item

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Rename-Item

Set-Item

Aliases
si

Command Type
Cmdlet

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Set-Item

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Set-Item

ItemProperty Commands

The following are examples of policies that you can create to support the execution of ItemProperty cmdlets in an interactive session or by using invoke-command.

Clear-ItemProperty

Aliases
clp

Command Type
Cmdlet

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Clear-ItemProperty

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Clear-ItemProperty
Copy-ItemProperty

Aliases
cpp

Command Type
Cmdlet

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Copy-ItemProperty

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Copy-ItemProperty

Get-ItemProperty

Aliases
gp

Command Type
Cmdlet

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Get-ItemProperty

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Get-ItemProperty

Move-ItemProperty

Aliases
mp

Command Type
Cmdlet

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Move-ItemProperty
Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Move-ItemProperty

New-ItemProperty

Command Type
Cmdlet

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command New-ItemProperty

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Get-WmiObject
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command New-ItemProperty

Remove-ItemProperty

Aliases
rp

Command Type
Cmdlet

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Remove-ItemProperty

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Remove-ItemProperty

Rename-ItemProperty

Aliases
rnp

Command Type
Cmdlet
**Policy Example (Enter-PSSession)**
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Rename-ItemProperty

**Policy Example (Invoke-Command)**
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Rename-ItemProperty

**Set-ItemProperty**

**Aliases**
sp

**Command Type**
Cmdlet

**Policy Example (Enter-PSSession)**
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Set-ItemProperty

**Policy Example (Invoke-Command)**
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Set-ItemProperty

**Location Commands**
The following are examples of policies that you can create to support the execution of Location cmdlets in an interactive session or by using invoke-command.

**Get-Location**

**Aliases**
gl, pwd

**Command Type**
Cmdlet

**Policy Example (Enter-PSSession)**
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Get-Location

**Policy Example (Invoke-Command)**
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Get-Location
Pop-Location

Aliases
popd

Command Type
Cmdlet

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Pop-Location

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Pop-Location

Push-Location

Aliases
pushd

Command Type
Cmdlet

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Push-Location

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Push-Location

Set-Location

Aliases
cd, chdir, sl

Command Type
Cmdlet

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Set-Location
Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Set-Location

Path Commands

The following are examples of policies that you can create to support the execution of Path cmdlets in an interactive session or by using invoke-command.

Join-Path

Command Type
Cmdlet

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Join-Path

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Join-Path

Convert-Path

Aliases
cvpa

Command Type
Cmdlet

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Convert-Path

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Convert-Path

Split-Path

Command Type
Cmdlet
Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Split-Path

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Split-Path

Resolve-Path

Aliases
rvpa

Command Type
Cmdlet

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Resolve-Path

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Resolve-Path

Test-Path

Command Type
Cmdlet

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Test-Path

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Test-Path

PSDrive Commands

The following are examples of policies that you can create to support the execution of PSDrive cmdlets in an interactive session or by using invoke-command.
Get-PSDrive

Aliases
gdr

Command Type
Cmdlet

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Get-PSDrive

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Get-PSDrive
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Get-WmiObject

New-PSDrive

Aliases
mount, rdr

Command Type
Cmdlet

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command New-PSDrive

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command New-PSDrive
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Get-WmiObject

Remove-PSDrive

Aliases
rdr

Command Type
Cmdlet
**Policy Example (Enter-PSSession)**

Add-PowerBrokerPolicy -Principal EXAMPLE\smith
  -Command Remove-PSDrive

**Policy Example (Invoke-Command)**

Add-PowerBrokerPolicy -Principal EXAMPLE\smith
  -Command Remove-PSDrive
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
  -Command Get-WmiObject

**PSProvider Commands**

The following are examples of policies that you can create to support the
execution of PSProvider cmdlets in an interactive session or by using
invoke-command.

**Get-PSProvider**

**Command Type**

Cmdlet

**Policy Example (Enter-PSSession)**

Add-PowerBrokerPolicy -Principal EXAMPLE\smith
  -Command Get-PSProvider

**Policy Example (Invoke-Command)**

Add-PowerBrokerPolicy -Principal EXAMPLE\smith
  -Command Get-PSProvider

**Additional Commands**

The following are examples of policies that you can create to support the
execution of additional cmdlets in an interactive session or by using invoke-
command.

**Get-Alias**

**Aliases**

gal

**Command Type**

Cmdlet

**Policy Example (Enter-PSSession)**

Add-PowerBrokerPolicy -Principal EXAMPLE\smith
  -Command Get-Alias
Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Get-Alias
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Get-Help
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command ForEach-Object
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Where-Object
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Select-Object

Set-ExecutionPolicy

Command Type
Cmdlet

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Set-ExecutionPolicy

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Set-ExecutionPolicy

Get-Process

Aliases
gps, ps

Command Type
Cmdlet

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Get-Process

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Get-Process

Get-Service

Aliases
gsv
Command Type
Cmdlet

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Get-Service

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Get-Service

Get-Eventlog

Command Type
Cmdlet

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Get-Eventlog

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Get-Eventlog

Function Examples
The following are examples of policies that you can create to support the execution of functions in an interactive session or by using invoke-command.

C:

Aliases
Set-Location C:

Command Type
Function

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Set-Location
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command C: -CommandType Function
Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Set-Location
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command cd\ -CommandType Function

cd..

Aliases
Set-Location ..

Command Type
Function

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Set-Location
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command cd.. -CommandType Function

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Set-Location
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command cd\ -CommandType Function

cd\n
Aliases
Set-Location \\n
Command Type
Function

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Set-Location
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command cd\ -CommandType Function

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command Set-Location
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
-Command cd\ -CommandType Function
help

Command Type
Function

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command more -CommandType Function
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command more.com -CommandType Application
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Out-String
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command help -CommandType Function

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command more -CommandType Function
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command more.com -CommandType Application
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Out-String
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command help -CommandType Function
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command get-help

mkdir

Command Type
Function

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command New-Item
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command mkdir -CommandType Function

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command New-Item
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command mkdir -CommandType Function
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command get-WmiObject
Application Examples

The following are examples of policies that you can create to support the execution of applications in an interactive session or by using invoke-command.

**ipconfig**

**Command Type**
Application

**Policy Example (Enter-PSSession)**
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Out-String
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command ipconfig.exe -CommandType Application

**Policy Example (Invoke-Command)**
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Out-String
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command ipconfig.exe -CommandType Application

**ping**

**Command Type**
Application

**Policy Example (Enter-PSSession)**
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Out-String
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command ping.exe -CommandType Application

**Policy Example (Invoke-Command)**
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Out-String
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command ping.exe -CommandType Application

**notepad**

**Command Type**
Application
Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Out-String
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command notepad.exe -CommandType Application

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Out-String
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command notepad.exe -CommandType Application

calc

Command Type
Application

Policy Example (Enter-PSSession)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Out-String
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command calc.exe -CommandType Application

Policy Example (Invoke-Command)
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command Out-String
Add-PowerBrokerPolicy -Principal EXAMPLE\smith
   -Command calc.exe -CommandType Application
Appendix B: PowerBroker Servers Events

There are two types of PowerBroker Servers events: Proxy Host events and Run Host events.

Proxy Host Events

The following events are applicable to the Proxy Host. The event source is listed as PBWS Proxy.

- Authentication events: 10001-10002
- Authorization events: 10003-10004
- Security Token Service (STS) events: 10005-10006
- Proxy Host session events: 10007-10009
- ASP .NET event: 10010
- Web Services-Management (WS-Management) event: 10011

Event 10001, PBWS Proxy

This event is generated when the PowerBroker Servers Proxy Host has successfully authenticated an administrator from an authorization request initiated from a Submit Host.

Symbolic Name: PRX_AUTH_REQUEST_SUCCEEDED

Message

A <AuthenticationType> authentication type request completed successfully.

Subject:

Account Name: <AdministratorUserName>
Account Domain: <AdministratorDomain>

Submit Host:

Submit Host Name: <SubmitHostName>
Submit Host IP Addresses: <SubmitHostIPAddresses>

Target Host:

Target Host Name: <RunHostName>
Target Host IP Addresses: <RunHostIPAddresses>
Event 10002, PBWS Proxy

This event is generated when the PowerBroker Servers Proxy Host fails to authenticate an administrator from an authorization request initiated from a Submit Host.

Symbolic Name: PRX_AUTH_REQUEST_FAILED

Message
A request could not be authenticated.

Subject:

Authentication Type: <AuthenticationType>
Account Name: <AdministratorUserName>
Account Domain: <AdministratorDomain>

Submit Host:

Submit Host Name: <SubmitHostName>
Submit Host IP Addresses: <SubmitHostIPAddresses>

Target Host:

Target Host Name: <RunHostName>
Target Host IP Addresses: <RunHostIPAddresses>

Reason:

<AuthenticationDecision>

Event 10003, PBWS Proxy

This event is generated when the Policy Decision Point (PDP) component of the PowerBroker Servers Proxy Host authorizes a request to create a remote Windows PowerShell session.

Symbolic Name: PRX_AUZ_REQUEST_SUCCEEDED

Message
Authorization completed successfully.

Subject:

Account Name: <AdministratorUserName>
Account Domain: <AdministratorDomain>

Submit Host:
Submit Host Name: <SubmitHostName>
Submit Host IP Addresses: <SubmitHostIPAddresses>

Target Host:
Target Host Name: <RunHostName>
Target Host IP Addresses: <RunHostIPAddresses>

**Event 10004, PBWS Proxy**

This event is generated when the Policy Decision Point (PDP) component of the PowerBroker Servers Proxy Host fails to authorize a request to create a remote Windows PowerShell session.

Symbolic Name: PRX_AUZ_REQUEST_FAILED

**Message**
Authorization failed.

Subject:
Account Name: <AdministratorUserName>
Account Domain: <AdministratorDomain>

Submit Host:
Submit Host Name: <SubmitHostName>
Submit Host IP Addresses: <SubmitHostIPAddresses>

Target Host:
Target Host Name: <RunHostName>
Target Host IP Addresses: <RunHostIPAddresses>

Reason:
Policy Decision: <PolicyDecision>

**Event 10005, PBWS Proxy**

This event is generated when the request to the Security Token Service (STS) completes successfully and elevated credentials are provided.

Symbolic Name: PRX_STS_REQUEST_SUCCEEDED

**Message**
STS request completed successfully.

Subject:
Account Name: <AdministratorUserName>
Account Domain: <AdministratorDomain>

Account Whose Credentials Will Be Used:
Account Name: <PowerAdministratorUserName>
Account Domain: <PowerAdministratorDomain>

Submit Host:
Submit Host Name: <SubmitHostName>
Submit Host IP Addresses: <SubmitHostIPAddresses>

Target Host:
Target Host Name: <RunHostName>
Target Host IP Addresses: <RunHostIPAddresses>

**Event 10006, PBWS Proxy**
This event is generated when the request to the Security Token Service (STS) refuses to provide elevated credentials.

Symbolic Name: PRX_STS_REQUEST_FAILED

**Message**
STS request failed.

Subject:
Account Name: <AdministratorUserName>
Account Domain: <AdministratorDomain>

Submit Host:
Submit Host Name: <SubmitHostName>
Submit Host IP Addresses: <SubmitHostIPAddresses>

Target Host:
Target Host Name: <RunHostName>
Target Host IP Addresses: <RunHostIPAddresses>

Reason:
STS Decision: <STSDecision>
Event 10007, PBWS Proxy

This event is generated when the Windows PowerShell session is created.
Symbolic Name: PRX_PS_SESSION_CREATED

Message
PowerShell session has been created.
Subject:

Account Name: <AdministratorUserName>
Account Domain: <AdministratorDomain>

Submit Host:

Submit Host Name: <SubmitHostName>
Submit Host IP Addresses: <SubmitHostIPAddresses>

Target Host:

Target Host Name: <RunHostName>
Target Host IP Addresses: <RunHostIPAddresses>

PowerShell:

Shell ID: <PowerShellSessionID>

Event 10008, PBWS Proxy

This event is generated when a Windows PowerShell session could not be created.
Symbolic Name: PRX_PS_SESSION_FAILED

Message
PowerShell session could not be created.
Subject:

Account Name: <AdministratorUserName>
Account Domain: <AdministratorDomain>

Submit Host:

Submit Host Name: <SubmitHostName>
Submit Host IP Addresses: <SubmitHostIPAddresses>

Target Host:
Target Host Name: <RunHostName>
Target Host IP Addresses: <RunHostIPAddresses>

Reason:
Error Code: <ErrorCode>
Error Description: <ErrorDescription>

**Event 10009, PBWS Proxy**

This event is generated when the Windows PowerShell session is terminated.

Symbolic Name: PRX_PS_SESSION_ENDED

**Message**
PowerShell session has been terminated.

Subject:
Account Name: <AdministratorUserName>
Account Domain: <AdministratorDomain>

Submit Host:
Submit Host Name: <SubmitHostName>
Submit Host IP Addresses: <SubmitHostIPAddresses>

Target Host:
Target Host Name: <RunHostName>
Target Host IP Addresses: <RunHostIPAddresses>

PowerShell:
Shell ID: <PowerShellSessionID>

**Event 10010, PBWS Proxy**

This event is generated when an ASP.NET unexpected error occurs.
Symbolic Name: PRX_ASPNET_UNEXPECTED_ERROR

**Message**
An unexpected error occurred.

Subject:
Account Name: <AdministratorUserName>
Account Domain: <AdministratorDomain>

Submit Host:

Submit Host Name: <SubmitHostName>
Submit Host IP Addresses: <SubmitHostIPAddresses>

Target Host:

Target Host Name: <RunHostName>
Target Host IP Addresses: <RunHostIPAddresses>

PowerShell:

Shell ID: <PowerShellSessionID>

Reason:

Error Code: <ErrorCode>
Error Description: <ErrorDescription>

Event 10011, PBWS Proxy

This event is generated when an unexpected error occurs while a Web Services-Management (WS-Management) message is being processed.

Symbolic Name: PRX_WSMAN_UNEXPECTD_ERROR

Message

An unexpected error occurred while processing a WS-Management message.

Subject:

Account Name: <AdministratorUserName>
Account Domain: <AdministratorDomain>

Submit Host:

Submit Host Name: <SubmitHostName>
Submit Host IP Addresses: <SubmitHostIPAddresses>

Target Host:

Target Host Name: <RunHostName>
Target Host IP Addresses: <RunHostIPAddresses>

PowerShell:

Shell ID: <PowerShellSessionID>
Reason:

   Error Code: <ErrorCode>
   Error Description: <ErrorDescription>

Run Host Events

The following events are applicable to Run Hosts. The event source is listed as PBWS Authorization Manager.
   - Authorization management events: 20002-20003, 20006
   - Other events: 20005

Event 20002, PBWS Authorization Manager

This event is generated when the Microsoft Authorization Manager denies the execution of a command.

Symbolic Name: PWS_ACCESS_DENIED_BY_SYSTEM

Message
PowerShell Command execution denied by Microsoft Authorization Manager.

Subject:
   Account Name: <AdministratorUserName>
   Account Domain: <AdministratorDomain>

Submit Host:
   Submit Host Name: <SubmitHostName>
   Submit Host IP Addresses: <SubmitHostIPAddresses>

Elevated User Credentials:
   Account Name: <PowerAdministratorUserName>
   Account Domain: <PowerAdministratorDomain>

Target Host:
   Target Host Name: <RunHostName>
   Target Host IP Addresses: <RunHostIPAddresses>

PowerShell:
   Shell ID: <PowerShellSessionID>
   Command Name: <CommandName>
Event 20003, PBWS Authorization Manager

This event is generated when the PowerBroker Servers Authorization Agent denies the execution of a command.

Symbolic Name: PWS_ACCESS_DENIED_BY_PBWS

Message
PowerShell Command execution denied by PowerBroker Authorization Manager.

Subject:
Account Name: <AdministratorUserName>
Account Domain: <AdministratorDomain>
Submit Host:
Submit Host Name: <SubmitHostName>
Submit Host IP Addresses: <SubmitHostIPAddresses>
Elevated User Credentials:
Account Name: <PowerAdminisratorUserName>
Account Domain: <PowerAdministratorDomain>
Target Host:
Target Host Name: <RunHostName>
Target Host IP Addresses: <RunHostIPAddresses>
PowerShell:
Shell ID: <PowerShellSessionID>
Command Name: <CommandName>
Command Type: <CommandType>
PDP:
Decision: <PolicyDecision>

Event 20005, PBWS Authorization Manager

This event is generated when an unexpected error occurs.

Symbolic Name: PWS_UNEXCEPTD_ERROR
Message
An unexpected error occurred.

Subject:

Account Name: <AdministratorUserName>
Account Domain: <AdministratorDomain>

Submit Host:

Submit Host Name: <SubmitHostName>
Submit Host IP Addresses: <SubmitHostIPAddresses>

Target Host:

Target Host Name: <RunHostName>
Target Host IP Addresses: <RunHostIPAddresses>

PowerShell:

Shell ID: <PowerShellSessionID>

Reason:

Error Description: <ErrorDescription>

Event 20006, PBWS Authorization Manager

This event is generated when the PowerBroker Servers Authorization Agent detects an attempt to execute a command.

Symbolic Name: PWS_EXECUTE_COMMAND

Message
PowerBroker Authorization Manager detected an attempt to execute a Command.

Subject:

Account Name: <AdministratorUserName>
Account Domain: <AdministratorDomain>

Submit Host:

Submit Host Name: <SubmitHostName>
Submit Host IP Addresses: <SubmitHostIPAddresses>

Elevated User Credentials:
Account Name: <PowerAdministratorUserName>
Account Domain: <PowerAdministratorDomain>

Target Host:
  Target Host Name: <RunHostName>
  Target Host IP Addresses: <RunHostIPAddresses>

PowerShell:
  Shell ID: <PowerShellSessionID>
  Command Name: <CommandName>
  Command Type: <CommandType>
Appendix C: Troubleshooting

The following topics provide information about troubleshooting the PowerBroker Servers Proxy Host, the Run Host, the PowerBroker Servers Authorization Agent, and the PowerBroker Servers AuthHelper Service.

Troubleshooting the Proxy Host

To turn on logging for the PowerBroker Servers Proxy Host, edit the following file on the Proxy Host:

C:\Program Files\BeyondTrust\PowerBroker for Windows Server\Proxy Server\WSMAN\web.config

Modify the <BT.PowerBroker.Logging> section to meet your needs. By default, all logging except event logs is commented out. To enable the required logger, uncomment the corresponding entry. You can also modify the logger to meet your needs.

The following are some examples of changes you can make:

- Change the log file location:
  <traceListener name="Performance"
    source="c:\Logs\Proxy\perflog.txt" />

- Change the level of logging:
  <traceListener name="Performance"
    source="c:\Logs\Proxy\perflog.txt" switchValue="Off"/>

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Logs all events</td>
</tr>
<tr>
<td>Off</td>
<td>Turns off logging for all events</td>
</tr>
<tr>
<td>Critical</td>
<td>Logs only events of the following type: System.Diagnostics.TraceEventType.Critical</td>
</tr>
<tr>
<td>Error</td>
<td>Logs only events of the following types: System.Diagnostics.TraceEventType.Critical System.Diagnostics.TraceEventType.Error</td>
</tr>
<tr>
<td>Warning</td>
<td>Logs only events of the following types: System.Diagnostics.TraceEventType.Critical System.Diagnostics.TraceEventType.Error System.Diagnostics.TraceEventType.Warning</td>
</tr>
<tr>
<td>Verbose</td>
<td>Logs only events of the following types: System.Diagnostics.TraceEventType.Critical</td>
</tr>
</tbody>
</table>
## Value | Description
---|---
System.Diagnostics.TraceEventType.Warning | System.Diagnostics.TraceEventType.Warning
System.Diagnostics.TraceEventType.Information | System.Diagnostics.TraceEventType.Information

**ActivityTracing**
Logs only events of the following types:
- System.Diagnostics.TraceEventType.Stop
- System.Diagnostics.TraceEventType.Start
- System.Diagnostics.TraceEventType.Suspend
- System.Diagnostics.TraceEventType.Transfer
- System.Diagnostics.TraceEventType.Resume

---

### Troubleshooting the Run Host: PowerBroker Servers Authorization Agent

To aid in troubleshooting, you can customize the Run Host logging. To do so, edit the following configuration file on the Run Host:

C:\Program Files\BeyondTrust\PowerBroker for Windows Server \RunHost\BT.PowerBroker.PowerShell.Authorization.dll.config

Modify the `<BT.PowerBroker.Logging>` section to meet your needs. By default, all logging except event logs is commented out. To enable the required logger, uncomment the corresponding entry. You can also modify the logger to meet your needs.

The following are some examples of changes you can make:

- **Change the log file location or file name.** To do so, modify the corresponding entry. For example:

```
<traceListener name="TraceLog"
    source="c:\Logs\AuthMan\tracelog.txt"
    switchValue="Off" />
<traceListener name="PDP"
    source="c:\Logs\AuthMan\pdp.log" switchValue="Off" />
<traceListener name="Engine"
    source="c:\Logs\AuthMan\engine.txt" switchValue="Off" />
<traceListener name="Events" source="PowerBroker Servers Authorization Manager" />
```

- **Change the log level by modifying the switchValue attribute of the desired log to one of the following:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Logs all events</td>
</tr>
<tr>
<td>Off</td>
<td>Turns off logging for all events</td>
</tr>
</tbody>
</table>
### Value Description

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| Critical | Logs only events of the following type:  
  System.Diagnostics.TraceEventType.Critical |
| Error    | Logs only events of the following types:  
  System.Diagnostics.TraceEventType.Critical  
  System.Diagnostics.TraceEventType.Error   |
| Warning  | Logs only events of the following types:  
  System.Diagnostics.TraceEventType.Critical  
  System.Diagnostics.TraceEventType.Error  
  System.Diagnostics.TraceEventType.Warning |
| Activity | Logs only events of the following types:  
  System.Diagnostics.TraceEventType.Critical  
  System.Diagnostics.TraceEventType.Error  
  System.Diagnostics.TraceEventType.Warning  
  System.Diagnostics.TraceEventType.Information |
| Verbose  | Logs only events of the following types:  
  System.Diagnostics.TraceEventType.Critical  
  System.Diagnostics.TraceEventType.Error  
  System.Diagnostics.TraceEventType.Warning  
  System.Diagnostics.TraceEventType.Information  
  System.Diagnostics.TraceEventType.Verbose |
| ActivityTracing | Logs only events of the following types:  
  System.Diagnostics.TraceEventType.Stop  
  System.Diagnostics.TraceEventType.Start  
  System.Diagnostics.TraceEventType.Suspend  
  System.Diagnostics.TraceEventType.Transfer  
  System.Diagnostics.TraceEventType.Resume |

### Troubleshooting the Run Host: PBWS Service

To turn on logging for PBWS Service, stop the service with the following command:

```plaintext
net stop "PBWS Service"
```

If this command does not work, open Task Manager and kill the BT.PowerBroker.AuthHelper.exe process.

After the service is stopped, edit the following configuration file on the Run Host:

```plaintext
C:\Program Files\BeyondTrust\PowerBroker for Windows Server\RunHost\BT.PowerBroker.AuthHelper.exe.config
```

Modify the `<BT.PowerBroker.Logging>` section to meet your needs. By default, all logging except event logs is commented out. To enable the required logger, uncomment the corresponding entry. You can also modify the logger to meet your needs. For more information, see examples in “Troubleshooting the Run Host: PowerBroker Servers Authorization Agent,” page 70.

Save the file, start the service, and look for clues in the log file.